KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Mine, Interact, Learn, Repeat

Interactive Pattern-based Data Exploration

Vladimir Dzyuba

Supervisors:
Prof. dr. Luc De Raedt
Dr. Matthijs van Leeuwen
(Leiden University, the Netherlands)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

June 2017

Mine, Interact, Learn, Repeat

Interactive Pattern-based Data Exploration

Vladimir DZYUBA

Examination committee:
Prof. dr. ir. Adhemar Bultheel, chair
Prof. dr. Luc De Raedt, supervisor
Dr. Matthijs van Leeuwen, supervisor
(Leiden University, the Netherlands)
Prof. dr. Bart Baesens
Prof. dr. Bettina Berendt
Prof. dr. Siegfried Nijssen
(Université Catholique de Louvain, Belgium)
Prof. dr. Alexandre Termier
(Université de Rennes 1, France)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

June 2017

Abstract

In many fields, the rapid growth of the amount of available data has created the need for automated tools to assist analysts in understanding these data and discovering useful knowledge in them. *Pattern mining* is a well-studied knowledge discovery task, which aims at providing concise, comprehensible descriptions of coherent regions in the data. Many variations of pattern mining have been proposed in the literature, together with even more algorithms to efficiently mine the corresponding patterns. However, the vast majority of these methods do not adapt their results to the goals and interests of a particular analyst, which makes pattern mining inaccessible to non-expert users and hampers its adoption as a practical data exploration tool.

In this thesis, we investigate algorithmic approaches to interactive pattern mining, where an analyst only needs to provide feedback with respect to intermediate results, which is then used to steer the mining process towards *subjectively* interesting results (patterns). We frame this problem as an interactive mining and learning loop that can be paraphrased by the formula "*Mine, interact, learn, repeat.*" The main contributions of this thesis are the techniques that implement individual steps of this loop.

The first contribution is an algorithm to learn user preferences for patterns from *ordered feedback* and methods to minimize the amount of user feedback required to learn an accurate user model. The second contribution is a flexible pattern sampling algorithm, which supports a wide range of pattern constraints and sampling distributions and generates diverse, representative collections of patterns on demand. The third contribution is an end-to-end interactive pattern mining algorithm that combines preference learning with "anytime" mining by sampling.

Experiments demonstrate that the techniques presented in this thesis perform well in a variety of pattern mining tasks and thus are promising building blocks for practical interactive data exploration systems.

Beknopte samenvatting

De snelle toename in de hoeveelheid beschikbare data heeft de nood gecreëerd naar geautomatiseerde tools die analisten ondersteunen wanneer zij de data wensen te doorgronden en hieruit nuttige informatie wensen te halen. *Pattern mining* is een goed bestudeerde *knowledge discovery* taak die tot doel heeft om begrijpbare beschrijvingen van coherente delen van de data (patronen) te genereren. Het leeuwendeel van *mining* methodes past echter zijn resultaten niet aan aan de interesses en doelen van de analist, wat tot gevolg heeft dat *pattern mining* ontoegankelijk wordt voor niet-experten en dat de inzet van pattern mining als een praktische data exploratie tool in het gedrang komt.

In deze thesis onderzoeken we algoritmische benaderingen van interactieve *pattern mining*, waar de analist enkel feedback dient te geven met betrekking tot tussentijdse resultaten, dewelke dan gebruikt wordt om het algoritme te sturen naar subjectief interessante resultaten (patronen). We kaderen dit probleem als een interactief zoeken en leren loop die kan geparafraseerd worden door de formule: "*Zoek, interacteer, leer, herhaal.*" De voornaamste bijdragen van deze thesis zijn technieken die de individuele stappen van deze loop implementeren.

De eerste contributie is een algoritme voor het leren van de voorkeuren van de gebruiker voor bepaalde patronen van geordende feedback, en methodes om de totale hoeveelheid feedback nodig om een accuraat model te leren, te minimaliseren. De tweede bijdrage is een flexibel *sampling* (steekproef) algoritme, dat een wijde range aan *constraints* (beperkingen) en waarschijnlijkheidsverdelingen ondersteunt, en dat op vraag diverse, representatieve collecties van patronen kan genereren. De derde contributie is een compleet *end-to-end* interactief *pattern mining* algoritme dat het leren van gebruikersvoorkeuren combineert met *anytime* zoeken via *sampling*.

Experimenten tonen aan dat de technieken voorgesteld in deze thesis, goed presteren in een verscheidenheid aan *mining* taken, en bijgevolg veelbelovende bouwstenen zijn voor praktische interactieve data exploratie systemen.

Acknowledgements

As a PhD student, I haven't *read* many dissertations: they are rarely cited, and most of their content is available as individual papers, which are much easier to find. Nevertheless, if I stumbled upon one, I would almost always browse the acknowledgements section to try to understand how another person experienced their PhD. Now it's my turn to write one.

First and foremost, a PhD is a research affair. I would like to thank Luc for providing me with the opportunity to do research. Along the way, Luc always reminded me to keep an eye on the big picture and showed how to draw useful connections between seemingly unrelated things. Matthijs was there from Day 0: he supervised my master thesis (which later turned into my first paper; see Section 2.5). I could always rely on Matthijs to provide an idea, a pointer to relevant literature, a lesson in writing or making slides, time for writing and reading our joint papers, general advice on how to survive a PhD, an enthusiasm boost, and many other things. These four and a half years would have been awfully harder without Matthijs! Siegfried also provided a great deal of advice, ranging from getting all the scientific details right to low-level matters, e.g., grammar aspects of using the SVM acronym. Many of the ideas explored in this dissertation were extensively discussed with him.

I would like to thank Adhemar for chairing the private and public defences and Alexandre, Bart, and Bettina for agreeing to join the jury. Their broad knowledge, challenging questions, and sharp suggestions allowed me to improve the manuscript considerably. Erik Duval was a member of my supervision committee; I had had several long and illuminating conversations with him before he tragically passed away...

I would like to thank Fonds Wetenschappelijk Onderzoek (FWO) for the financial support of my research via the project "Instant Interactive Data Exploration", together with colleagues from the University of Antwerp. Meetings and brainstorming sessions with Bart, Jilles, Sandy, Emin, Koen, and others

were always a good source of ideas and discussions. I learned a lot through this and other collaborations I've been part of: with Mario, Thomas, Björn, and Bo during a research visit to Bonn as well as with Antti.

The DTAI Sports Analytics Lab was another fun research experience. We could passionately argue about football players or basketball news, while working on papers and other groundbreaking projects[1]. I would like to thank Anton, Wannes, Joris, Tim, Toon, Tom, and Vincent for making it so rewarding; Jesse for heading the lab, and Jan for all his effort that went into coming up and pushing the projects, while being the go-to source for the latest sports analytics advances. Moreover—also in relation to sports—Jan helped me to start *quizzen*, which I had been missing since I came to Belgium and which I enjoy a lot, even despite the language barrier.

One of the e-mails that I received on the first day of my PhD was a teaching schedule: I was supposed to teach my first exercise session in a few weeks. It was a bit scary initially, but over the years I came to enjoy teaching and supervising thesis students. Thus, I would like to thank Hendrik and Jesse for their guidance and my fellow TAs Nima, Jérôme, Kurt, and Behrouz.

I would like to thank the secretariat of the doctoral school and the department for helping with oh-so-complex administrative matters. The biggest "thank you" goes to Karin.

I would never be able to reach the research goals on my own, without support. Luckily, it has always been easy to find in our group; many people were much more than just colleagues, who could turn even the worst things into memories. Case in point: laundry. This was the time Benjamin and I would discuss everything, from life at the department to programming to history, and so on. Besides laundry, there were numerous coffee breaks, a unique chicken with potatoes, and many more things I remember fondly. Irma was a great source of inspiration and energy. A lot of plans would never have been realized, if it weren't for Irma: studying Dutch and going to the gym to name just two. All in addition to a steady everyday supply of jokes, memes, and good conversations. Moreover, a couple of Irma and Guy's "interventions" definitely helped me make good decisions at important points.

For the whole duration of my PhD, I stayed in the same office. I was lucky to have many great officemates, but Thanh was certainly the greatest. I could always ask for help and support or discuss anything, science or just life. Dinners at his apartment with his family were *very* cozy. Sharing the office with Albrecht,

[1] `http://whomightwin.it/euro2016/`

Ondrej, and (for the longest period) Tias helped me learn a lot about doing research. Vincent has brought a lot of fresh energy into the room.

There were plenty good moments inside and outside of the department: ALMA, coffee (tea) breaks, house parties, Secret Santas, movies, the trip to Westvleteren, hiking in Bouillon, watching football, eating a zabbaglione; all thanks to the colleagues: Anna, Antoine, Anton, Evgeniya, Jessa, Irina & Sergey, Kurt, Leander, Nikolina & Sebastijan, Pedro, Samuel, Toon, and many others. Long conversations with Behrouz and Francesco hold a special place in my heart. Guy helped me deal with the impostor syndrome and gave good research advice (see Chapter 4). I was always looking forward to the weekly CW-Voetbal matches and (sometimes) dinners in De Spuye, so I would like to thank Alex, Bogdan, Constantin, Emad, Dominique, Geert, Gijs, José, Keith, Kristof, Matthias, Niraj, Oleksandr, Rinde, Tuur, and others for the great atmosphere. A special "thank you" goes to Jonas, who was the recipient of numerous questions about the last phases of the PhD in the final months.

I had moved to Belgium a year before I started my PhD. I was very lucky to meet people who helped me find a second home here: Natasha A., Natasha P. & Maarten, Karina & Shamil, Lev, and Anton; Daryna and Monika; Dima, Dimitrios, Ivan, Lillian, and Vera; and Hernan and David. Nastya's move to Brussels was a pleasant suprise, followed by cozy dinners, entertaining walks, and adventures involving furniture. Lesha Ivanov's invitation to Eindhoven for a New Year party came at a timely moment. Finally, Eric & Martine literally provided me with a home for the duration of my PhD; I couldn't find a better studio, even though I tried.

Maintaining old relationships over distance may be harder than creating new ones in a foreign place. I'm afraid I haven't been good it, but luckily, my friends have. Conversations by e-mail, Skype, group chats, or during the infrequent meetings in Russia or elsewhere (London, Boston, Madrid, or Vienna) have been an important source of support. Alisa, Kirill, and Andrey; Kirill, Lesha, and Toma; Dasha and Liza; Nikita, Anton I.; Lev, Nastya; and others – it might not have been obvious, but I appreciate your support very very much.

The same applies to my family: at certain points, they had to deal with me being low on energy and absent-minded, which, I can imagine, required a lot of strength and patience. I am very very grateful that they had enough and even more to support me, give advice, and simply welcome me at home or wherever else we would meet. I would like to thank my parents Galina and Andrey; my sister Katya, Vova, Andrey, and Anna; my aunt Lora and uncle Vova; my cousin Petya, Olga, and Vasilisa; of course, my grandmother Lyuba, and others for sticking with me through weird and cheerful times alike. Спасибо огромное!!!

Leen—among many other things—snapped me out of that absent-mindedness and provided a much needed energy boost. In fact, most research covered in this dissertation—Chapters 4 and 5—was completed and published after that. (This is a very robotic statement, but you suspect I am a robot anyway.) I'm not sure I would've been able to do it otherwise. Thank you `xoxo`

I was taught to end my papers or presentations with clear take-away messages. So, to any PhD student, who is reading this for amusement or out of curiosity (who else would still be reading at this point?): my PhD was a bumpy ride, but there have always been people to support me along the way, intellectually or emotionally. So, look around for those.

Vladimir Dzyuba
Seogwipo, South Korea
May 2017

Contents

List of Symbols

$\mathbf{1}_S$	Multiplicity of multiset S
$\lvert S \rvert$	Cardinality (size) of set S
$\binom{n}{m}$	Number of ways to choose an unordered subset of m elements from a set of n elements
$[\cdot]$	Iverson bracket
$\succ$	Preference relation, e.g., $p_1 \succ p_2$ implies that user preferes p_1 to p_2
ς	Cell sampling strategy
λ	Minimal pattern description length
$\varphi(\cdot)$	Pattern quality measure
ρ	Spearman rank correlation
θ	Minimal pattern frequency threshold
$\chi^2(\cdot)$	Quality measure *chi-squared*
$\mathcal{A}$	Set of all attributes
A	Dataset attribute
$\mathcal{C}$	Set of constraints on patterns
C_p	Cover of pattern p
$\mathcal{D}$	Dataset
$\mathrm{Dom}(A)$	Domain of attribute A
E_U	User effort, i.e., the costs of providing feedback

$freq\,(p)$	Frequency of pattern p, i.e., $\lvert C_p \rvert$
$h\,(\cdot)$	Learned quality measure (*hypothesis* about user preferences)
$\mathcal{I}$	Set of all items
$\mathcal{L}$	Pattern language
p	Pattern
$\mathcal{P}$	Set of patterns
$q\,(p)$	Selection predicate; returns `true`, if p is an interesting pattern
Q	Query, i.e., a set of patterns shown to the user
Q^*	Feedback item, i.e., a ranked query
R	Pattern ranking
R^*	Pattern ranking by subjective interestingness
$\hat{R}$	Learned pattern ranking
Rec_k	Recall at k
t	Dataset tuple, e.g., a transaction in binary data
$\mathcal{T}$	Set of all transaction indices
U	Information about user, e.g., user feedback w.r.t. patterns
X	Instance space

List of Abbreviations

ACFI Sampler based on **a**pproximate **c**ounting of **f**requent **i**temsets

CP Constraint programming

CSP Constraint satisfaction problem

DE Discounted error

DSSD Diverse Subgroup Set Discovery algorithm

EDA Exploratory data analysis

EF EFlexics

F Constraint set $\{minfreq(\cdot)\}$

FCL Constraint set $\{minfreq(\cdot), closed, minlen(\cdot)\}$

FIM Frequent itemset mining

GF GFlexics

JS *Jensen-Shannon* divergence

LRW Sampler based on **l**attice **r**andom **w**alks

MCMC Markov Chain Monte Carlo

MMR Maximum Marginal Relevance heuristic

RDD Relevance, Diversity, Density heuristic

SCD Stochastic Coordinate Descent algorithm

SD Subgroup discovery

TS Two-step sampler

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we investigate algorithmic approaches to knowledge discovery via interactive pattern mining. This chapter briefly introduces the problem domain, draws parallels with similar challenges in information retrieval, and provides an overview of the contributions of the thesis and its general structure.

1.1 Knowledge discovery & data mining

The technological advances in data collection, storage, and processing occurring at an increasing pace since the late 1980s resulted in a wide range of theoretical and practical challenges regarding utilizing these data and led to the emergence of the research field of *knowledge discovery in databases* (KDD)[1], which is defined as [53]:

> The process of automatically identifying models from massive observational databases that are valid, novel, useful, and understandable.

The term *data mining* generally refers to the crucial step in the knowledge discovery process that follows data pre-processing and consists in using an algorithm to build a model of a particular type. The seminal work by Agrawal et al.—which resulted in several of the most cited papers in computer science [2, 4, 3]—kickstarted *pattern mining*, one of the most prominent subfields of KDD and data mining [149, 1]. Informally[2], a pattern is a statement in a formal

[1] KDD is tightly related to the active field of *data science* [114].

[2] See Chapter 2 for a formal description of pattern mining.

	Louise	Emma	Marie	Olivia	Nora	*Points scored?*
1	*Plays*	P	P	*Rests*	P	+
2	P	P	R	P	P	−
3	P	P	P	R	R	+
4	P	R	R	P	R	−
5	R	P	R	P	P	−

Table 1.1: A toy dataset about a basketball game. The first five attributes indicate whether a particular player was playing or having rest in a given game segment, whereas the final attribute indicates whether the team succeeded in scoring points in that segment.

language that concisely describes a subset of a given dataset. Pattern mining algorithms aim at providing comprehensible descriptions of coherent regions in the data. The term "*mining*" metaphorically refers to finding "*nuggets* of knowledge" (e.g., patterns) in *raw* data.

For example, Table 1.1 shows an example dataset, where each row corresponds to a segment in a basketball game, five attributes indicate whether a corresponding team member was playing or resting during a given segment, and the last attribute indicates whether the team succeeded in scoring points during the segment. Consider the following pattern: $p = \{\text{Emma}, \text{Marie}\}$ ($|C_p| = 2$, $\left|C_p^+\right| = 2$, $\left|C_p^-\right| = 0$). The first part (p) is the *description* of a pattern, whereas the numbers in brackets characterize the subset matching the description, or *covered* by it, i.e., the total number of segments as well as the number of successful and unsuccessful segments ($|C_p|$, $\left|C_p^+\right|$, and $\left|C_p^-\right|$ respectively). This pattern corresponds to the following nugget of knowledge: "*if both Emma and Marie are playing, the team scores points more often*", i.e., in 100% of such segments (2 out of 2) vs. 40% of all segments in the whole data (2 out of 5). It might be interesting to the coach of the team, as it might indicate that Emma and Marie should be playing together more often.

Pattern languages are typically designed to be *understandable*, i.e., easy to interpret even for non-experts. Therefore, from the KDD perspective, the primary challenge in pattern mining is to identify the (few) patterns that are *novel* (or *interesting*) and *useful* (or *actionable*). Very early in pattern mining development, Silberschatz and Tuzhilin pointed out that these notions are *subjective*, i.e., they do not only depend on the data at hand, but also on the knowledge and goals of the analyst (the user of the algorithm) [128]. However, despite this observation, research in pattern mining initially focused heavily on the efficiency of the proposed algorithms, forgoing the subjective aspects [60].

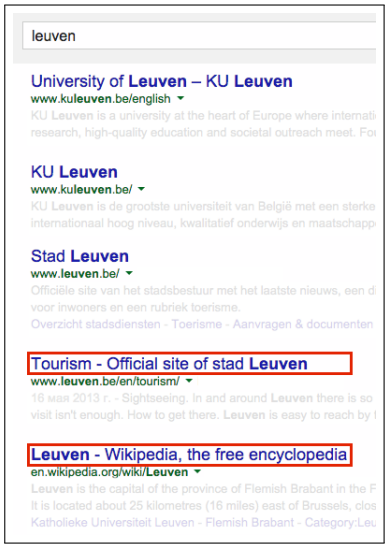
This trend was curbed in recent years, when various researchers started proposing ways to take the user into account in the pattern mining process either by incorporating a formal model of the user knowledge and interests into their algorithms, or by actively involving her in pattern discovery [138]. In this thesis, we build upon the parallels with the issues that the field of *information retrieval* faced in the early phases of its development and the techniques that were developed by that research community to address these issues. (The initial steps in this direction were taken by Xin et al. [150] and Rüping [121].)

1.2 Information retrieval

Information retrieval (IR) is concerned with obtaining information relevant to the information needs of a user from a collection of information resources. Search engines that provide access to the (primarily textual) information on the World Wide Web are prototypical IR applications. Arguably, certain theoretical and practical challenges in IR are similar to those of KDD: 1) the user's knowledge of the domain and/or the underlying resources may be limited; 2) there is no precise description of the desired result; 3) furthermore, the desired results vary depending on the user; and 4) user sessions tend to be open-ended. Indeed, in both IR and KDD a user seeks to address her information needs with the help of an algorithm; a dataset can be likened to a document collection, which potentially contains answers to the user's needs in the form of either relevant documents or interesting patterns. The main task of the algorithm is to return said documents or patterns.

In IR research, the solutions to these challenges were motivated by the most common form of output: an ordered list of documents. Given a query, an IR algorithm is expected to put the most relevant documents at the top of the list. An increasing amount of user behavior data, e.g., which documents are eventually clicked on by users, motivated the development of a wide range of *learning to rank*, or *preference learning*, algorithms, which use these data to learn models that order query results.

For example, Figure 1.1 shows the search engine output for the query '`leuven`'. A person interested in visiting Leuven as a tourist will likely skip the first three results (the university webpages and the official page of the city administration) and click on the links to the page of the tourism office and the Wikipedia page. From these actions, the search engine infers that this user would prefer that the clicked pages appear higher in the ranking, e.g., P4 $\succ$ P1 stands for "Page 4 (Tourism) should be ranked above Page 1 (University of Leuven)." A preference learning algorithm uses these data (example preferences) along with features of



Pages & clicks	Inferred ranking	Inferred preferences
P1	P4 P5	
P2		P4 ≻ P1 P5 ≻ P1
P3	P1	P4 ≻ P2 P5 ≻ P2
P4 ←	P2	P4 ≻ P3 P5 ≻ P3
P5 ←	P3	

Figure 1.1: Search engines use data on clicks of users and preference learning algorithms to improve the ranking of query results.

documents (e.g., word occurrences) and users (e.g., click history) to personalize the ranking of query results. This is an instance of a *machine learning* task, where the examples of observed cases and their outcomes (e.g., users interacting with search results) are used to build, or *learn*, a model that predicts outcomes of future cases (e.g., ranking documents by the likelihood of being clicked).

Preference learning has been highly successful in IR [73, 94], as evidenced by the pervasiveness of search engines powered by it. This prompted us to investigate whether these techniques can be applied in knowledge discovery and pattern mining. In the following section, we further elaborate on the similarities and differences between these fields and describe our research objectives.

1.3 Towards interactive pattern mining

Imagine a typical search engine session from the user's perspective: she types a simple query into the search bar and receives a list of documents as the result; she scans it and clicks on the most relevant ones or updates her query. Despite the complexity of the underlying machinery, the user experience is intuitive and can be mastered even by novice computer users. Our main research objective is to design algorithms that enable a similar user experience in knowledge discovery

with pattern mining, where a non-expert user could get acquainted with the dataset at hand by exploring the pattern space and moreover, have the capacity to steer the exploration according to her personal interest and analysis goals, akin to updating the query. This would make pattern mining more accessible to a wide range of users, who have basic data analysis skills yet strong expertise in their domain ("citizen data scientists" [113]).

In particular, the success of learning from user behavior in IR indicates the importance of the *subjective* perspective, where the output of an algorithm depends on (the model of) the user that is consuming it, as opposed to the *objective* perspective, where the results only depend on the input, e.g., in IR, on the query and the document collection. The same observation holds for KDD. Recall the basketball example from Section 1.1:

- The coach of the team might be interested in player combinations that correlate to successful actions in order to determine the players who should receive more playing time.
- The coach of the opponent is interested in the combinations correlated with *unsuccessful* actions to find the team's weak spots.
- The interests of a journalist might be completely different than those of coaches. Furthermore, they might be very diverse, as she might want to write several different stories based on the data.

Covering these scenarios with state-of-the-art objective pattern mining approaches is problematic, as the algorithms must be carefully tuned, which requires substantial expertise that domain analysts do not have. In other words, although building blocks for complex KDD machinery already exist, unlike in IR, they are not easily accessible to non-experts. Hence, pattern mining can benefit from borrowing personalization techniques from IR. However, a number of differences preclude the straightforward transfer of methods.

First, in KDD, there is typically no initial query that provides a strong indication of the user's interests and goals. Second, various kinds of side information that are widely used in IR (e.g., word semantics, language models, links between documents, and others) are unavailable in pattern mining. Finally, IR systems are typically used by a large number of users simultaneously, providing a much larger amount of the user behavior data compared to pattern mining that focuses on a single user. On the other hand, unlike IR systems, pattern mining algorithms can "synthesize" new documents, i.e., generate patterns, fairly freely. These differences suggest that compared to IR, a much tighter interaction between the user and the mining system is necessary, which raises new research challenges as well as opens up new opportunities.

The approach we assume consists in *alternating* between mining patterns and learning a model of the user interests. It has been studied in the context of post-processing the output of objective mining algorithms [122, 150] and directly searching for patterns [121], with the last two approaches using IR-inspired methods. However, these studies focused on specific tasks, and a thorough general investigation has been missing. To this end, in this thesis we propose a high-level interactive pattern mining framework, which views the problem as an interactive mining and learning loop that can be paraphrased by the formula "*Mine*, *interact*, *learn*, *repeat*."

First, given a dataset, an initial set of patterns is mined and presented to the user who interacts with them and provides feedback with respect to their subjective interestingness. Afterwards, a model of user interests is learned from this feedback and, most importantly, used to guide the pattern search, when mining is repeated. Thus, over a number of iterations, an algorithm learns what patterns the current user is interested in and mines them.

This procedure entails a number of important design choices: what patterns to show to the user, what kind of feedback to let her provide, how to build a user model from the feedback, how to use the model to mine (subjectively) more interesting patterns, and others. These are the issues we investigate in this thesis, building upon approaches from IR and adapting them to pattern mining specifics.

1.4 Contributions

In this thesis we focus on the following research questions, which concern the major steps within the proposed framework:

Q1 (Mine) What properties are essential for a mining algorithm as a part of an interactive mining framework?

Q2 (Interact) What kind of feedback is required to organize convenient and effective interaction between a user and the mining process?

Q3 (Interact) How can the total effort required from the user to identify subjectively interesting patterns be minimized?

Q4 (Learn) Can existing machine learning techniques leverage the user feedback to learn an accurate user model?

Q5 (Repeat) How can the learned user models be used to discover subjectively more interesting patterns?

The main contributions of this thesis regarding **Q1** are as follows (see Chapter 4):

- A new pattern sampling algorithm that builds upon the latest advances in weighted sampling in SAT and constraint programming for pattern mining that allows it to support a wide range of pattern constraints and quality measures, while providing strong performance guarantees; it ensures the *diversity* of mined patterns and enables *anytime* data exploration.
- An extension of Eclat [153], a specialized pattern mining algorithm, required for its integration into the proposed sampler.

Regarding **Q2** and **Q3**, the main contributions are as follows (see Chapter 3):

- An empirical evaluation of *ordered feedback*, where a user is requested to order patterns from the most interesting to the least interesting.
- A set of *active learning* heuristics that select patterns to be shown to a user with the goal of minimizing the total effort required to learn an accurate model of her interests.

Regarding **Q3** and **Q4**, the main contributions are an algorithm for learning subjective pattern quality measures from user feedback, which is based on the well-known *preference learning* paradigm (also knows as *learning to rank* in information retrieval), and two implementations thereof that rely on different learning techniques (see Chapters 3 and 5).

Regarding **Q5**, the main contributions are as follows (see Chapter 5):

- A class of pattern quality measures that admit efficient parameter learning (by means of preference learning) and sampling.
- An interactive pattern sampling algorithm that provides a genuine anytime implementation of the proposed framework; it leverages preference learning to learn parameters of a quality measure belonging to the aforementioned class and the pattern sampler introduced in this thesis to generate high-quality patterns in an anytime manner.

1.5 Structure of the thesis

Chapter 2 expands on the topics briefly covered in this chapter, presents relevant background information on pattern mining, and information retrieval

and outlines the high-level framework for interactive pattern-based data exploration. It contains a case study based on the following conference paper:

> V. Dzyuba, M. van Leeuwen. "Interactive discovery of interesting subgroup sets". In: *Proceedings of the 12th International Symposium on Intelligent Data Analysis (IDA '13)*. 2013, pp. 150–161

Chapter 3 introduces an approach to modeling and eliciting user interests, organizing her interaction with the algorithm, and learning a user model from her feedback. Thus, it addresses the *Interact* and *Learn* components of the framework. The approach is based on *active preference learning* and relies on the user providing *ordered feedback* with respect to patterns. In particular, we demonstrate how to learn pattern ranking functions using off-the-shelf preference learning algorithms and propose a number of active learning heuristics. Experiments demonstrate that preference learning has the capacity to learn accurate pattern rankings and that active learning heuristics help reduce the required user effort. Moreover, using learned ranking functions within search heuristics allows discovering novel high-quality patterns. The chapter consists of the research previously published in the following papers:

- V. Dzyuba, M. van Leeuwen, S. Nijssen, L. De Raedt. "Active preference learning for ranking patterns". In: *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '13)*. 2013, pp. 532–539
  Winner of the Best Paper Award (out of 299 submissions).
- V. Dzyuba, M. van Leeuwen, S. Nijssen, L. De Raedt. "Interactive learning of pattern rankings". In: *International Journal on Artificial Intelligence Tools* 23.06 (2014)

Chapter 4 studies the problem of pattern sampling, where deterministic search for patterns is replaced with principled randomized generation. Our interest in this problem is motivated by competitive performance of randomized methods in the experiments described in Chapter 3. We present a flexible pattern sampling algorithm that supports a wide range of sampling distributions and constraints on patterns and provides strong theoretical guarantees with respect to sampling accuracy and runtime. It builds upon the latest advances in weighted constrained sampling in SAT. This allows us to obtain an implementation of the *Mine* component with a number of desired properties, most importantly, diversity of results and the capacity for "anytime" data exploration. Experiments show that the proposed sampler is accurate and scales to large real-world datasets. The chapter is based on the following journal paper:

V. Dzyuba, M. van Leeuwen, L. De Raedt. "Flexible constrained sampling with guarantees for pattern mining". In: *Data Mining and Knowledge Discovery* (in press). arXiv: 1610.09263

Chapter 5 presents an interactive pattern discovery algorithm that puts together the ideas introduced in the previous chapters. It uses the sampler introduced in Chapter 4 for active learning and mining as well as preference learning for learning (subjective) pattern quality measures, which allow efficient sampling by taking advantage of the properties of the sampler. Thus, the algorithm covers all components of the "*Mine*, *interact*, *learn*, *repeat*" framework with approaches proposed in this thesis. The chapter is based on the following conference paper:

V. Dzyuba, M. van Leeuwen. "Learning what matters – Sampling interesting patterns". In: *Proceedings of the 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '17)*. 2017, pp. 534–546. arXiv: 1702.01975

In the concluding chapter, we summarize the thesis and discuss its limitations and opportunities for future work.

Chapter 2

Interactive pattern mining

We begin this chapter with a formal definition of pattern mining, an overview of the key challenges therein, and state-of-the-art techniques to address these challenges. Then we discuss user involvement in pattern mining and present our interactive pattern mining framework.

2.1 Pattern mining

Pattern mining aims to reveal structure in data in the form of patterns, with a strong emphasis on obtaining comprehensible descriptions. It targets *exploratory data analysis* (EDA) tasks [134, 111]. The primary goal of EDA is to provide the analyst with insights into the data. We contrast it with other well-known analysis tools, namely hypothesis testing and predictive modeling. In *statistical hypothesis testing*, the analyst seeks to confirm or disprove the validity of a single, precisely defined statistical relationship, typically using experimental data. The goal of EDA can be seen as (automatically) *suggesting* multiple hypotheses that would be reasonable to test, based on *observational* data.

The goal of *predictive modeling* is to use the historical data to build a model that predicts a value of interest in future, for example, a binary outcome in classification (e.g., "spam" or "not spam") or a numeric value in regression (e.g., a number of website visits). Predictive models necessarily focus on *global* modeling, as the situations, in which predictions would be made, are unknown at the model building time. As a result, the models that provide the best predictive performance often are not human-interpretable (cf. recent breakthroughs in *deep*

learning [91]). EDA is *descriptive*: its primary goal is to summarize relevant subsets of the data (i.e., *local* structure) in an interpretable form. Due to these contrasts, the continuous progress in hypothesis testing and predictive modeling does not directly lead to progress in EDA [38, 147]—the primary aim of this thesis. This topic has only recently begun to gain attention in confirmatory statistics [95], databases [123, 44], and other research communities [43].

The pattern mining task is commonly formalized as *theory mining* [98]. Given a dataset $\mathcal{D}$, a pattern language $\mathcal{L}$ defining subsets of $\mathcal{D}$, and a selection predicate q that determines whether a pattern $p \in \mathcal{L}$ describes an *interesting* subset of $\mathcal{D}$, the task is to find descriptions of all interesting subsets, i.e., $\{\, p \in \mathcal{L} \mid q\,(p, \mathcal{D}) \text{ is true} \,\}$. Therefore, a pattern consists of a description p, i.e., a formal statement in $\mathcal{L}$, and a corresponding cover $C_p = \{\, t \in \mathcal{D} \mid p\,(t) \text{ is true} \,\}$, i.e., (the indices of) the subset of $\mathcal{D}$ matching the description, or *covered* by it. We often omit the subscript p whenever it is clear from the context.

Pattern languages provide descriptions of subsets of a dataset and thus are specific to the particular type of data being mined. We assume that a dataset $\mathcal{D}$ is a bag (multiset) of homogeneous data instances, i.e., $\mathcal{D} = \{\, (j, t) \mid 1 \leq j \leq |\mathcal{D}|,\ t \in X \,\}$, where j is the (unique) instance index, t is the individual data instance, and X is the space of all possible instances. For example, the tabular form of data is one of the most common data types. Formally, let $\mathcal{A} = \{A_1, \ldots, A_{M-1}, A_M\}$ denote a set of attributes, where each attribute A_k has a domain of possible values $\mathrm{Dom}\,(A_k)$. The instance space $X = \mathrm{Dom}\,(A_1) \times \ldots \times \mathrm{Dom}\,(A_M)$ then comprises all possible tuples over $\mathcal{A}$, while datasets $\mathcal{D}$ are bags of tuples from X, i.e., a tuple t can occur multiple times in any given dataset. Attributes are single-valued, e.g., *binary* ($|\mathrm{Dom}\,(A)| = 2$), *nominal* ($\mathrm{Dom}\,(A)$ is a discrete set), or *numeric* ($\mathrm{Dom}\,(A) = \mathbb{R}$). The most common pattern language for tabular data consists of conjunctions of Boolean atoms (conditions) over individual attributes, e.g., $A_1 = a \ \wedge\ A_2 > 0$. Numeric attributes are often discretized in advance, e.g., by equal-width binning.

Fully binary, or 0/1, data, where all attributes in $\mathcal{A}$ are binary, are an important special case of tabular data. By convention, attributes are referred to as *items*, and data instances are referred to as *transactions*. Let $\mathcal{I} = \{1 \ldots M\}\,(= \mathcal{A})$ denote a set of items. The instance space is the powerset of $\mathcal{I}$; $X = 2^{\mathcal{I}}$. In other words, a dataset $\mathcal{D}$ is a bag of transactions over $\mathcal{I}$, where each transaction t is a subset of $\mathcal{I}$, i.e., $t \subseteq \mathcal{I}$. $\mathcal{T} = \{1 \ldots |\mathcal{D}|\}$ is the set of transaction indices. The pattern language also consists of sets of items; $\mathcal{L} = 2^{\mathcal{I}}$: an itemset p covers a transaction t, iff $p \subseteq t$.

If all attributes are of equal *ex ante* importance, the pattern mining task is considered *unsupervised*. On the contrary, if one or more attributes chosen by the user are of particular interest, the mining task is *supervised*. These attributes

are referred to as *target* attributes or *labels*, whereas all other attributes are referred to as *description* attributes. For instance, the basketball example in Section 1.1 is supervised, with "*points scored?*" as a target attribute and players as description attributes.

Subgroup discovery [83, 148] is a prototypical supervised mining task. It is an instance of supervised descriptive rule discovery [89]; see Atzmueller [7] for a recent survey. Subgroup discovery is concerned with finding subsets of a dataset that have a substantial deviation in a property of interest, compared to the entire dataset, e.g., an unusually high proportion of successful segments in the basketball example. Formally, let T denote the target attribute; thus, $\mathcal{A} \setminus T$ are description attributes. For brevity, here we assume a single binary target attribute (generalizations are possible [8, 140, 45]). Let $\mathrm{Dom}\,(T) = \{+, -\}$ and C^l (resp. $\mathcal{D}^l$) denote the set of tuples with label $l \in \mathrm{Dom}\,(T)$ in the subgroup cover (resp. in the entire dataset). A quality measure is a function $\varphi : \mathcal{L} \to \mathbb{R}$ that quantifies the difference between the label distribution in C and $\mathcal{D}$. The higher the value of φ, the more interesting is the pattern. Examples of quality measures include *sensitivity*, *specificity*, *weighted relative accuracy* ($WRAcc$), or *chi-squared* (χ^2):

$$Sensitivity\,(p) = \frac{|C^+|}{|\mathcal{D}^+|} \qquad Specificity\,(p) = 1 - \frac{|C^-|}{|\mathcal{D}^-|}$$

$$WRAcc\,(p) = \frac{|C|}{|\mathcal{D}|} \times \left(\frac{|C^+|}{|C|} - \frac{|\mathcal{D}^+|}{|\mathcal{D}|} \right)$$

$$\chi^2\,(p) = \sum_{l \in \{+,-\}} \frac{(|C|(|C^l| - |\mathcal{D}^l|))^2}{|C||\mathcal{D}^l|} + \frac{(|C|(|C^l| - |\mathcal{D}^l|))^2}{(|\mathcal{D}| - |C|)|\mathcal{D}^l|}$$

Tabular data are compactly represented with fixed-length tuples; to a certain extent, this also applies to pattern languages describing tabular data. More complex data types that cannot be represented in this way include sequence data, graph data, and relational data. While there is a wide range of pattern languages for these data and approaches to mine patterns in them, in this thesis we focus on tabular data and leave the extensions to complex data types to future work.

The final component of the pattern mining task is the selection predicate q. It defines the notion of pattern *interestingness*, which is task- and user-specific. However, efficiency is the competing consideration: ideally, q admits

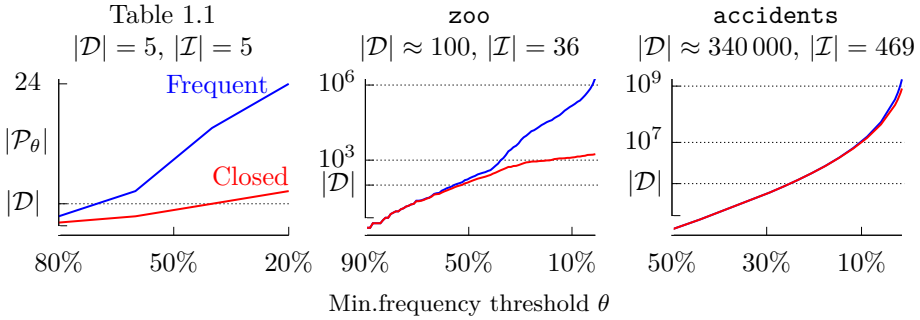


Figure 2.1: Correspondence between the minimal frequency threshold θ and the number of frequent patterns $\mathcal{P}_\theta$ for a number of datasets. For low frequency thresholds, the number of frequent patterns can exceed the size of the data by orders of magnitude. For large, high-dimensional datasets, condensed representations (here closed patterns) may fail to reduce the result set size.

efficient evaluation, i.e., enumeration of all patterns that satisfy q, that is, all interesting patterns. *Frequent pattern mining* [1] is the seminal and most well-studied pattern mining variant, where q involves a constraint on the minimal frequency of a pattern p, i.e., the minimal size of its cover C_p. Formally, $q(p) = freq(p) \geq \theta$, where $freq(p) = |C_p|/|\mathcal{D}|$ is the *frequency* of a pattern and θ is a user-provided frequency threshold. Enumerating all frequent patterns is typically highly efficient due to the *anti-monotonicity* property of the minimal frequency constraint, which states that all supersets of an infrequent pattern are also infrequent. This property allows pruning large parts of the search space. However, defining pattern interestingness solely via minimal frequency rarely matches task-specific needs [60]. The primary reason is *pattern explosion*, which we discuss in the following section.

2.2 Pattern explosion

The term "pattern explosion" refers to the rapid pace at which the number of frequent patterns grows as the minimal frequency threshold θ decreases; see Figure 2.1 for examples. Even in small datasets the number of frequent patterns for low θ can exceed the size of the dataset by orders of magnitude; in large datasets the number of frequent patterns is typically humongous. In other words, the size of the description of a dataset with patterns vastly exceeds the size of the dataset itself. Furthermore, from the KDD perspective, this cannot be easily alleviated by tuning the threshold θ: the most frequent patterns returned

Method	Selection predicate q	Pros	Cons
Frequent pattern mining	$freq\,(p) \geq \theta$	*Efficiency*	*Pattern explosion*
Condensed representations	$freq\,(p)$ is not computable from frequencies of $\mathcal{P} \setminus p$	Efficiency	Large result sets
Constraint-based mining	p satisfies (local) constraints $\mathcal{C}$	-Flexibility -Efficiency	Hard to tune
Top-k mining	$p \in$ top-k w.r.t. quality measure φ	-Flexibility -Efficiency	Redundancy
Pattern set mining	p is *different* from patterns in $\mathcal{P} \setminus p$	-Compactness -Diversity	High runtime costs
Pattern sampling	$P\,(q\,(p; \mathcal{D}) = \text{true}) \propto \varphi\,(p)$	-Anytime -Diversity	Low flexibility

Table 2.1: Overview of state-of-the art approaches to alleviating pattern explosion. $\mathcal{P}$ denotes the set of patterns that satisfy the selection predicate q.

for high thresholds, while valid, typically correspond to common knowledge and thus are neither novel nor useful [24]. If the user lowers the threshold, however, the pattern explosion occurs, and the task of identifying *the few* genuinely interesting patterns in huge pattern collections is not automated by frequent pattern mining algorithms and is left to the user.

Pattern explosion stems from the simplistic approach to pattern interestingness in frequent pattern mining: it considers any pattern that describes a sufficient number of instances interesting. However, most instances can be described (covered) by multiple patterns, only few of which reveal genuine, dataset-wide structural regularities. Other patterns are redundant, i.e., re-describe the same regularity using different statements or adding superfluous details, or are caused by spurious correlations, which become increasingly likely as the dimensionality of the data grows. Below we discuss the extensions to frequent pattern mining that aimed at addressing its flaws, namely *condensed representations*, *constraint-based mining*, *top-k mining*, *pattern set mining*, and *pattern sampling*; see Table 2.1 for a high-level summary.

Condensed representations Condensed representations [31] tackle pattern explosion by modifying q to include only the frequent patterns that are *not*

redundant to any other pattern in the returned pattern collection. For example, *closed* patterns [110] essentially eliminate contextual deterministic dependencies. To illustrate, assume that item C always occurs if items A and B occur, e.g., $freq\,(\text{itemset }\{AB\}) = 10$, $freq\,(\{C\}) = 8$, and $freq\,(\{ABC\}) = 8$. Then (the information provided by) the pattern $\{C\}$ is redundant to $\{AB\}$ and $\{ABC\}$, hence it should not be returned. A wide range of other condensed representations have been proposed, including *maximal* [9], *δ-free* [22], *non-derivable* [30], *self-sufficient* [146], and other representations, some of which are approximate, or *lossy*, i.e., do not allow reconstructing the set of all frequent patterns and their frequencies exactly.

However, even though many condensed representations can be mined efficiently, they still yield large result sets that cannot be processed by human analysts. Figure 2.1 shows the number of closed patterns; for example, for the large `accidents` dataset (see Chapter 4 for details) the number of closed patterns is virtually equivalent to the number of frequent patterns.

Constraint-based mining In constraint-based mining [107], the selection predicate q involves a variety of constraints on individual patterns, in addition to (or instead of) minimal frequency. It is tightly related to condensed representations: for example, non-closed patterns can be eliminated by adding a particular constraint [68]. Another example from itemset mining involves constraining the total cost of items in a pattern: $\sum_{i \in p} cost\,(i) \gtrless \omega$; it may be used to constrain the *length* of the pattern description ($cost\,(\cdot) \equiv 1$). In general, the study of constraints has been prominent in pattern mining: a wide range of constraint classes were investigated, including anti-monotonic constraints [3], convertible constraints [112], and others. As a result, generic mining systems that could freely combine various constraints were proposed [28, 20, 68].

Constraint-based mining offers flexibility in specifying which patterns are (not) interesting. Moreover, many constraints can be handled efficiently. However, in exploratory data analysis the exact specification of the mining task is often unknown in advance, therefore it is problematic for non-experts to choose and tune the constraint set. Moreover, its effect on the size of the result set is still non-transparent: too many or too few patterns are returned depending on the user choice.

Top-k mining In contrast to hard constraints, top-k mining [154] focuses on modeling soft preferences regarding pattern interestingness with *quality measures* [100, 59]. A quality measure $\varphi : \mathcal{L} \to \mathbb{R}$ is a function that for a given dataset $\mathcal{D}$ maps a pattern to a number; the larger the number, the more interesting is the pattern according to the measure φ. The selection predicate q is then

satisfied by k patterns with the highest values of φ, where k is a user-specified parameter: $q(p) = p \in \arg_k \max_{p' \in \mathcal{L}} \varphi(p')$.

The most basic quality measure is the frequency itself (i.e., we are interested in the most frequent patterns, regardless of the threshold). Subgroup discovery is an important top-k mining task; we list several examples of subgroup quality measures in Section 2.1. An example of an unsupervised quality measure in itemset mining is the *surprisingness* of the frequency of an itemset with respect to the independence model: $Surprisingness(p) = freq(p) - \prod_{i \in p} freq(\{i\})$.

Top-k mining does not suffer from pattern explosion by design, as only k patterns are returned. Nevertheless, it does not address one of the root causes of the explosion, namely the redundancy of result sets: top-k patterns typically contain many variations of the same theme, while many potentially interesting patterns fall outside the top-k and are thus completely ignored. This is a crucial drawback for many practical situations, where a quality measure is just an approximation of the genuine pattern interestingness [34].

Pattern set mining In order to tackle the redundancy, the selection predicate q must consider the entire result set, i.e., the interestingness of a particular pattern must depend on other returned patterns. This is the core idea underlying pattern set mining [25]. In supervised settings, where patterns can be ranked, e.g., top-k, pattern set mining often consists in pruning or penalizing patterns redundant to higher-ranked ones during post-processing [86, 85, 26] or directly within the search [140], e.g., using sequential covering that aims at reducing the overlap of covers of patterns in the result set [90].

KRIMP is a prominent unsupervised approach based on compression principles from information theory [143]. Another group of unsupervised approaches based on information theory iteratively selects patterns, whose properties are surprising given the previously seen patterns [97]. Boolean matrix factorization essentially computes a set of itemsets that compactly describes the data [103]. Alternatively, pattern set mining can be seen as satisfying *global* constraints that concern multiple patterns (or the entire result set) as opposed to *local* constraints on individual patterns in constraint-based mining [41, 80, 69].

Pattern set mining combines the advantages of condensed representations and top-k, as it returns small non-redundant high-quality patterns. However, it typically is computationally more intensive than the aforementioned methods. Thus, it is necessary to resort to heuristic methods [67, 70], which often require choosing the number of patterns in advance or return suboptimal results [141].

Pattern sampling All variants of pattern mining described above assumed that all patterns that satisfy the selection predicate q must be returned. In pattern sampling, a pattern is returned with a certain probability. The probability can be uniform [15], i.e., each pattern is equally likely to be sampled, or, mirroring top-k mining, proportional to a quality measure; the latter category includes methods based on Markov Chain Monte Carlo (MCMC) [72, 14] or special purpose procedures [17, 19]. Furthermore, zero probability can be assigned to the patterns that do not satisfy constraints of choice, e.g., minimal frequency.

Pattern sampling is a promising solution to pattern explosion The main benefits of sampling are 1) diversity of results in that patterns are independently sampled from different regions in the solution space, and 2) the *anytime* nature in that patterns can be sampled one by one and a growing representative (albeit incomplete) set of patterns can be generated and inspected at any time. However, existing approaches either are not flexible enough, i.e., they only support a limited number of combinations of constraints and sampling distributions, or require tuning and pre-processing in order to generate genuinely independent samples. This is one of the issues we tackle in this thesis (see Chapter 4).

In sum, a variety of approaches to alleviating pattern explosion have been proposed, each with its own advantages and disadvantages, as expected. However, all of them are *objective*, or user-agnostic, i.e., they assume that pattern interestingness depends exclusively on the data[1]; $q : \mathcal{L} \times \mathcal{D} \rightarrow \{\text{true}, \text{false}\}$. In the following section we discuss *subjective* approaches, which directly take the user into account.

2.3 User involvement in pattern mining

The objective perspective on pattern mining, while fruitful, cannot capture a wide range of knowledge discovery scenarios. Different analysts have different knowledge and goals and thus will prefer different results, even given the *same* data. Recall the basketball example from Section 1.1: there, if various analysts (coaches or journalists) tried to satisfy their individual information needs using an objective pattern mining system, they would need to choose and tune constraints, quality measures, heuristics, and their parameters, i.e., reason about algorithms rather than the data and their domain of expertise. Most algorithms are "black boxes" that require considerable data mining expertise to do that, which makes them inaccessible to non-experts (Figure 2.2).

[1] Although this is only partially true for constraint-based and top-k methods, as they provide flexibility in the choice of constraints and measures, most research on those approaches focused on efficiency in purely objective tasks.

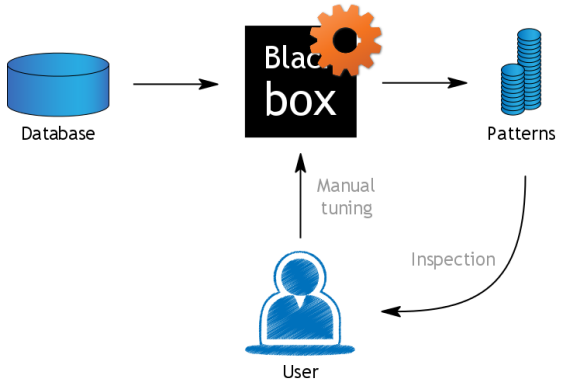


Figure 2.2: Objective mining algorithms are opaque to non-experts and typically require tedious manual tuning to tailor the results to the application at hand.

The *subjective* perspective on pattern mining assumes that pattern interestingness depends on the data as well as on the user, her knowledge, interests, and analysis goals: $q : \mathcal{L} \times \mathcal{D} \times U \to \{\text{true}, \text{false}\}$, where U is some kind of information about the user's interests. Even though it had emerged early in pattern mining development [82, 128], arguably, it has not received as much attention as the objective perspective [60, 138]. Below we list the examples of subjective approaches to pattern mining, which we categorize into two groups: 1) the approaches that build a user model before mining patterns and 2) interactive approaches that directly involve the user in the mining process.

Building user models Silberschatz and Tuzhilin [128] identified two independent aspects of pattern interestingness: *actionability* and *unexpectedness* (or *surprisingness*). The notion of actionability is typically domain-dependent. For example, in business applications, it may refer to comprehensibility and justifiability of a pattern from the domain perspective; operational efficiency, i.e., costs of establishing if a pattern applies to a new case (e.g., a client); profitability of exploiting a pattern, or compliance with regulations [81, 99].

Surprisingness of a pattern can be measured relative to a model of user beliefs that specifies the expected values of pattern characteristics, e.g., its frequency. The beliefs are strengthened or weakened by discovered patterns. Types of user models include association rules [108], Bayesian networks [75], constraints on data or pattern characteristics (e.g., column sums) [131, 40, 87], and others; see Kontonasios et al. [88] and Vreeken and Tatti [144] for extensive overviews. In particular, the approaches based on modeling beliefs as constraints on expected

values of statistics sparked the development of a family of methods based on the maximum entropy principle from information theory [39].

Interactive pattern mining A number of approaches based on modeling user's beliefs support *iterative* mining, where intermediate results are used to update the model and thus change the results at the following iterations. Nevertheless, they only accept input from the user *before* mining and then proceed in automatic mode. This reduces their flexibility and capacity to address various mining scenarios; cf. the basketball examples above. *Interactive* methods that directly involve the user in the mining process allow her to steer it towards subjectively more interesting regions in the pattern space and thus have the capacity to build more accurate and detailed user models.

The majority of interactive approaches are fairly recent. Sahar [122] proposed an algorithm for interactive filtering of pattern collections, where for each pattern, the user provides two judgements on whether the pattern is 1) true (as opposed to "not always true") and 2) interesting, which are then used to filter the collection. In a small user study, the author observed that the vast majority of rules are evaluated as "not always true, not interesting" or "true, not interesting", highlighting the importance of direct user involvement. Bhuiyan and Hasan [12] proposed an MCMC-based interactive itemset sampler, which allows the user to "like" or "dislike" patterns. This feedback is used to update weights of individual items; the product of the weights for items in an itemset determines its sampling probability. Graphical tools for pattern mining, such as MIME [64], SIREN [56], or VIPER [139], mine an initial set of patterns and let the user modify or remove patterns interactively. However, they essentially provide facilities for manual pattern space exploration with minimal assistance.

Most recent interactive approaches are based on *preference learning*, also known as *learning to rank*. This idea was first put forward by Xin et al. [150] and Rueping [121] and then independently developed by Boley et al. [16], Bhuiyan and Hasan [13], and in the research described in this thesis. In the following section we describe the principles underlying these algorithms and introduce a unifying framework for interactive pattern mining.

2.4 Mine, Interact, Learn, Repeat

The core idea behind interactive mining systems is to alternate between mining and updating the user model. We propose to frame this problem as an *interactive learning and mining loop* that consists of three major steps:

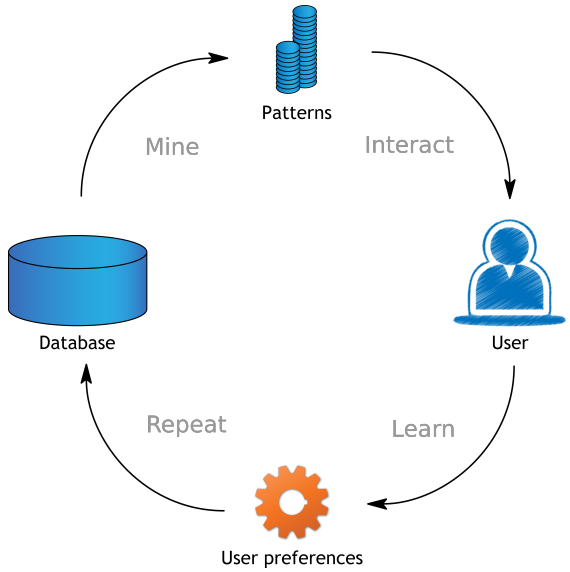


Figure 2.3: "Mine, interact, learn, repeat": interactive pattern mining as an interactive mining and learning loop.

Mining patterns In this step, a mining algorithm is used to find patterns that are to be presented to the user. It is crucial that the mining algorithm allows some form of subjective input. In general, this input is initially empty, and mining is essentially objective. However, in later iterations, an increasingly precise model of user interests becomes available, which allows for discovering subjectively more interesting patterns.

Interacting with the user The patterns are presented to the user, who can freely explore and inspect them. Meanwhile, feedback is elicited from the user, either implicitly or explicitly. The feedback to the learning system should be simple in order for the system to be accessible to non-data mining experts, yet at the same time should convey enough information about the user's interests.

Learning user-specific pattern interestingness The elicited feedback is used to build and improve the model of the user interests, i.e., to identify what makes patterns interesting to the user. Most importantly, the model is used in the next mining iteration, so that the effort required to achieve the analysis goals is minimized. In this thesis, we tackle this step as a *machine learning* problem, hence the term "learning."

This generic loop can be paraphrased by the adage *"Mine, interact, learn, repeat"* (Figure 2.3). Naturally, each step of the loop entails a number of important design choices and related research questions:

Mine How to model subjective, user-specific pattern interestingness?
How to incorporate a user model into pattern mining?

Interact What patterns to show to the user?
What kind of feedback to request from the user?

Learn How to learn a user model from the feedback?

Repeat How to enable fast, anytime data exploration?
How to update the model incrementally?
When to terminate the loop?

Formal problem statement Table 2.2 provides a summary of the state-of-the-art in interactive pattern mining (not including the approaches proposed in this thesis) organized according to the proposed framework. It illustrates the large amount of flexibility with respect to the design choices posed by the proposed framework. In the sequel we provide a formal definition of the interactive pattern mining task and a particular perspective that we take in this thesis.

The interactive pattern mining task comprises two principal subtasks. The primary subtask corresponds to the "*Interact*" and "*Learn*" steps and consists in eliciting user feedback and learning a formal model of the user's interests. The secondary task corresponds to the "*Mine* " step and involves using this model to mine novel patterns that are subjectively interesting to the user (according to the learned model).

In this thesis we take the perspective that aims at learning to rank patterns according to their interestingness to the current user by means of learning a user-specific subjective pattern quality measure. Formally, let $\mathcal{D}$ denote a dataset, $\mathcal{L}$ a pattern language, $\mathcal{C}$ a (possibly empty) set of constraints on patterns, and $\succ$ the unknown subjective pattern preference relation of the current user over $\mathcal{L}$, i.e., $p_1 \succ p_2$ implies that the user considers pattern p_1 subjectively more interesting than pattern p_2:

Problem 1 (Learning)**.** *Given $\mathcal{D}$, $\mathcal{L}$, and $\mathcal{C}$, dynamically collect user feedback U with respect to patterns in $\mathcal{L}$ and use U to learn a (subjective) pattern interestingness function $h : \mathcal{L} \to \mathbb{R}$ such that $h(p_1) > h(p_2) \Leftrightarrow p_1 \succ p_2$.*

Problem 2 (Mining)**.** *Given $\mathcal{D}$, $\mathcal{L}$, $\mathcal{C}$, and h, mine a* compact *set of* interesting *patterns $\mathcal{P}_h$.*

Table 2.2: Overview of interactive mining approaches organized according to the "Mine, interact, learn, repeat" framework. The approaches marked with * were developed simultaneously with the methods proposed in this thesis and independently of them.

Method	*Mine*	*Interact*	*Learn*
Sahar [122]	Post-processing	Pair (*true?*, *interesting?*)	Filter
Xin et al. [150]	Post-processing	Ordered feedback	Preference learning
Rueping [121]	Standard	Ordered feedback	Preference learning
Bhuiyan and Hasan [12]	MCMC-based sampling	"Like" or "dislike"	Multiplicative weight updates
OCM* [16]	Switching algorithms	Ordered feedback	Preference learning
PRIIME* [13]	Standard	Graded feedback (ratings)	Softmax regression

The precise mining problem statement may mimic top-k mining, where the goal is to maximize the total interestingness of patterns in a k-set, i.e., $\mathcal{P}_h = \operatorname{argmax}_{\mathcal{P} \in \mathcal{L}^k} \sum_{p \in \mathcal{P}} h(p)$, or pattern set mining, where a measure of pattern set diversity (see Chapter 5 for an example) is optimized along with the total interestingness.

Similar to search engines, the choice of constraints as well as mining and learning techniques is the task of the mining system designer. This sophisticated machinery is hidden from the user, who only interacts with patterns.

In the following chapters we investigate various solutions to these problems, starting with methods inspired by learning to rank from information retrieval and proceeding to augment them with pattern mining-specific features. Each chapter contains further discussion of relevant related work, in particular, interactive pattern mining in Chapter 3; constraint-based mining, top-k mining, pattern set mining, and pattern sampling in Chapter 4; and interactive pattern sampling in Chapter 5.

2.5 Preliminary research

In this section, we briefly describe a preliminary research project that led us to draw parallels between techniques in pattern mining and information retrieval. The aim of this research was to develop an algorithm for subgroup set mining that could take knowledge and interests of its user into account with the purpose of steering the search towards subjectively interesting results. To this end, we proposed IDSD, an interactive version of the DSSD algorithm [140]

DSSD (*Diverse Subgroup Set Discovery*) is based on levelwise beam search: at each level $l \geq 1$, all frequent subgroups of length-l are generated and ordered by the values of a chosen quality measure φ descending. Only the top-w subgroups (the *beam*) are used for further refinement, i.e., selected to generate candidates for level $l + 1$, where w is the *beam width* parameter. Unlike the standard beam search, DSSD employs various diverse subgroup set selection heuristics to select diverse beams: instead of simply retaining the top-w, it greedily selects w high-scoring patterns that are different from each other, e.g., in terms of their descriptions or covers. Maintaining the diversity of candidate subgroups *during the search* increases the diversity of the final result set.

The proposed interactive approach, dubbed IDSD (*Interactive Diverse Subgroup Discovery*), augments DSSD with interactive beam selection: at each level, a user is allowed to inspect subgroups in the current beam and *like* or *dislike* them. The feedback serves two purposes: 1) pruning the beam by removing the disliked patterns and 2) re-weighing an *objective* subgroup quality measure φ based on the similarity of a given pattern to the evaluated ones:

$$\varphi'(p) = \varphi(p) \times \frac{1 + \sum_{q \in \text{liked}} \sigma(p, q)}{1 + \sum_{r \in \text{disliked}} \sigma(p, r)}$$

where $\sigma : \mathcal{L} \times \mathcal{L} \to [0, 1]$ is a pattern similarity measure, e.g., based on comparing descriptions or covers.

Thus, the objective measure essentially becomes *subjective*. The overall effect amounts to steering the search towards patterns that are more similar to the ones "liked" by the user. The experimental evaluation comprised quantitative and qualitative studies. In the former, the feedback was emulated, mirroring a user who considers few high-scoring subgroups common knowledge and dislikes subgroups that are similar to those. Experiments show that the proposed

Section 2.5 is based on the conference paper "Interactive discovery of interesting subgroup sets" [46].

Subgroup description	Size $\lvert C_p \rvert$	Quality $\varphi = WRAcc$
$opp\_def\_reb = Low \wedge opponent \neq ATL \wedge thabeet \neq Plays$	219	0.0692
$opp\_def\_reb = Low \wedge opponent \neq ATL$	222	0.0689
$opp\_def\_reb = Low \wedge opponent \neq ATL \wedge ajohnson \neq Plays$	222	0.0689
$opp\_def\_reb = Low \wedge opponent \neq PHI \wedge thabeet \neq Plays$	225	0.0685
$opp\_def\_reb = Low \wedge opponent \neq PHI$	228	0.0682

a) Without interaction (objective) – DSSD

Subgroup description	Size	Quality
$crawford \neq Plays \wedge matthews = Plays$	290	0.0328
$hickson = Plays$	143	0.0219
$crawford \neq Plays \wedge hickson = Plays$	63	0.0211
$matthews = Plays \wedge hickson = Plays$	99	0.0163
$matthews = Plays \wedge pace < 88.518$	303	0.0221

b) With interaction (subjective) – IDSD

Table 2.3: Comparison of the top objective subgroups obtained with DSSD and the top subjective subgroups obtained as a result of an interactive session with IDSD. The latter subgroup set is more diverse and interesting to the expert.

approach successfully eliminates undesired (combinations of) conditions from the result set and ensures its diversity.

The (informal) qualitative study involved a case study in sports analytics, where a sports journalist analyzed basketball data. In these data, each tuple corresponded to a game segment, and the attributes represented players and segment statistics. The binary target attribute indicated whether the *offensive rating*, a well-known basketball performance measure, of the team in question was higher than average; see the original paper [46] for the full dataset description.

We used $WRAcc$ as the base objective quality measure. Table 2.3 compares *a*) the top-5 subgroups returned by DSSD using the cover-based diverse beam selection heuristic and default parameter settings (see Van Leeuwen and Knobbe [140] for details) with *b*) the results of an interactive session, where the analyst used IDSD . The objective results suffer from a number of issues described in the previous sections. First, the results are clearly redundant, i.e., diversity could not be attained with the default parameter settings: together, the subgroups describe only a quarter of game segments (231 out of 923). Second, the descriptions of the top objective subgroups do not contains novel or useful information. For example, it is a trivial fact for experts that poor defensive

rebounding by an opponent ($opp\_def\_reb = Low$) makes scoring easier and thus correlates with high offensive rating; the *absence* of reserve players Thabeet and A. Johnson is not informative either.

The top subjective subgroups are considerably more diverse; together, they cover more than a half of the dataset (512 out of 923 segments). Furthermore, even though their objective quality as measured by $WRAcc$ is considerably lower, they are more interesting to the expert: for instance, Crawford, Matthews, and Hickson were key players, who played often. Obtaining these results required limited effort: 18 evaluations, 7 "likes" and 11 "dislikes." In contrast, discovering these subgroups in the output of DSSD would require inspecting more than a thousand subgroups (the lowest rank by $WRAcc$ among them is 1049).

The results above originate from a good example of a successful interactive session. In other sessions the domain expert deemed the results unsatisfactory due to various reasons: the search space was pruned too eagerly, or positive and negative evaluations were not properly balanced, or their effect was unintuitive. In sum, although the proposed approach to elicit and process user feedback is ad-hoc and has evident drawbacks, it has shown that it is capable of improving the subjective quality of results with limited effort from the user. Furthermore, it suggests the ingredients that we consider essential for an interactive pattern mining system: 1) seamless integration of interaction and learning *within* the mining process; 2) ranking patterns by interestingness; 3) user feedback with respect to compact intermediate results; and 4) using the feedback to adjust pattern rankings by means of manipulating a quality measure. The search for principled methods to use these ingredients for interactive pattern mining motivated us to explore related ideas in information retrieval, e.g., *preference learning*, which we use in Chapter 3 to learn subjective pattern ranking functions.

Chapter 3

Interactive learning of pattern rankings

3.1 Introduction

In this chapter, we propose a concrete instance of the interactive learning and mining loop that is based on the notion of *pattern rankings*. We model user-specific interestingness as a *total order over patterns*. That is, for any two patterns from a given pattern language, one of the two is deemed more interesting than the other. This results in a ranking over patterns, such that the most interesting patterns are ranked highest. It is such a pattern ranking that we would like to learn through interaction with the user.

To achieve this, we make the following assumptions about the user:

1. A user has an implicit preference between any pair of patterns, which does not change during a particular analysis session;
2. The costs of eliciting the complete preference relation, or pattern ranking, i.e., expressing it in any analytical or other form, are prohibitively high;

This chapter is based on the journal article "Interactive learning of pattern rankings" [50], which is the extended version of the conference paper "Active preference learning for ranking patterns" [49].

3. For any single pair of patterns, however, the user can accurately identify which of the two she prefers, i.e., which pattern she considers more interesting.

Our first main contribution is a *generic algorithm for the interactive learning of pattern rankings*, which are represented by means of *ranking functions*. That is, we employ a preference learning algorithm to infer a ranking function that can score any pattern in the pattern language considered. The absolute scores provided by this function are unimportant, but the relative scores define the ranking over the complete pattern space. Ranking functions are functions over a feature representation of the patterns, so that feature relevance can be learned from user feedback.

Feedback elicited from the user amounts to rankings of small sets of patterns, to which we will refer as *queries* in this paper. Executing a query implies that the system selects a few patterns, presents these to the user, and asks the user to rank them. By generalizing from the pattern rankings provided by the user to such queries, a subjective pattern interestingness measure is learned. We propose and evaluate a number of *active query selection* methods, which aim to minimize the feedback –and thus effort– required from the user, while ensuring that accurate ranking functions are learned.

When mining the initial patterns, no knowledge about the user is available. Therefore, the user can select an objective interestingness measure based on her prior beliefs. This interestingness measure determines the initial source ranking; the closer this ranking is to the subjective target ranking that is to be learned, the easier the learning task becomes. When a pool of patterns has been mined, queries can be selected and presented to the user. The ranking function is then updated based on feedback provided by the user, and then the learned ranking function is used to mine novel, hopefully more interesting patterns. The second main contribution is *the application of the proposed approach in the context of frequent itemset mining and subgroup discovery.*

The remainder of this chapter is organized as follows. Section 3.2 describes related work in detail. Then, Section 3.3 describes a toy example to illustrate how our framework interactively learns pattern rankings. Section 3.4 introduces our framework for learning pattern rankings, consisting of both a problem definition and detailed algorithm description, after which Section 3.5 discusses how to use the learned ranking functions for mining new patterns.

Section 3.4 presents the extensive experimental evaluation, for both frequent itemset mining and subgroup discovery. In order to perform a principled and objective evaluation of our methods, we emulate user preferences over patterns using several existing interestingness measures. The results show that the

algorithm is able to learn accurate pattern rankings, and that query selection heuristics help reduce the amount of input required for learning. Moreover, the learned ranking functions generalize well and allow discovering novel high-quality patterns when used for mining. After that, we round up with a discussion and conclusions in Sections 3.7 and 3.8.

3.2 Related work

In this section we describe work that is most closely related to ours, on the topic of subjective interestingness and interactive pattern mining on one hand, and on the topic of preference learning on the other hand. The discussion of the former is divided into three groups: 1) specifying a model of user interests in advance; 2) using user feedback to directly influence the search procedure; and 3) learning an explicit model of user interests.

Modeling user interests The approach by Jaroszewicz et al. [75] allows a user to specify a Bayesian network that represents beliefs about the data-generating process. The algorithm then finds *surprising attribute sets*, i.e., attribute sets for which the discrepancy between the expected and observed frequencies is larger than a certain threshold. The search is based on efficiently computing (or approximating) a large number of marginal distributions. A user can then manually update the Bayesian network, i.e., her beliefs, based on the inspected patterns, and subsequently repeat the mining process. One disadvantage of this approach is that it does not avoid the pattern explosion, at least not without tuning a threshold.

De Bie [39] has developed a general framework for exploratory data analysis that uses information theory to formalize subjective interestingness as surprisingness with respect to certain prior beliefs. Different types of prior beliefs can be used, for example, expected frequencies of individual items. Given these prior beliefs, a Maximum Entropy distribution is fit to represent the expected data. This distribution is then used to quantify how informative the pattern is, given the beliefs. The framework lends itself well to iterative data mining: starting from a model based solely on prior beliefs, one can look for the subjectively most interesting pattern, which can then be added to the model. Hence, the next discovered pattern will automatically be substantially different from the previous one; this helps avoiding redundancy. A disadvantage of this framework is that it currently only allows for scoring pre-mined pattern collections, but not for directly mining high-scoring patterns.

Interactive search Bhuiyan et al. [12] proposed IPM, an interactive technique that is based on Markov Chain Monte Carlo sampling of frequent itemsets. It is similar to IDSD described in Section 2.5. User interests are modeled via a scoring function that is a product of weights of individual items; the probability of sampling an itemset is proportional to its score. In each iteration, a user inspects a small set of sampled itemsets and provides feedback by *liking* or *disliking* them. This feedback is then used to update the weights: the weights of items comprising liked (resp. disliked) itemsets are increased (resp. decreased). As a result, itemsets that are more similar to liked ones become more likely to be sampled and presented to the user.

Learning models of user interests Preference learning has been previously used to identify interesting patterns in an interactive manner. Xin et al. [150] investigated learning a user-specific ranking of frequent patterns (primarily itemsets and sequences). A clustering-based method similar to information retrieval approaches is used to select patterns for feedback. However, they only consider a specific learning target based on the discrepancy between the expected and observed supports of a pattern, and they do not use the learned functions to search for novel patterns.

Rueping [121] demonstrated the feasibility of learning subgroup rankings and applying learned ranking functions to discover high-quality subgroups. However, Rueping does not discuss active learning aspects and uses a custom variant of the learner and data modifications that are specific to subgroup discovery. Therefore, it cannot be straightforwardly generalized to other pattern mining tasks, as we do in this chapter.

The *One Click Mining* system (OCM) [18] is a generic system for interactive data mining that is not restricted to a single pattern type. That is, contrary to our system, it considers multiple types of patterns at once. Essentially, it learns two types of preferences simultaneously. On one hand, it uses a multi-armed bandit strategy to learn which mining algorithms discover the patterns that are most positively evaluated by a user. For example, that the user prefers subgroups with a particular target to itemsets. These preferences are used to allocate computation time to the different algorithms. On the other hand, co-active learning is used to learn a pattern ranking function from *implicit* user feedback, i.e., the actions performed by the user in the graphical user interface. Conceptually, this system is very similar to the one proposed in this chapter, using similar techniques. Still, there are a number of key differences. One such difference is that OCM only mines objectively interesting patterns, i.e., the learned ranking function is not used in the mining phase. Another difference is that although we show that our framework is generic enough to deal with

different types of patterns, we choose to focus on one mining task at a time, mainly for reasons of transparency and ease of use for the user.

Negrevergne et al. [104] proposed *dominance programming*, a declarative language that allows one to explicitly specify pairwise preferences between patterns, which are then used to find a complete set of non-dominated patterns. Only modeling objective preferences has been studied so far; learning models from example preferences is an open research question.

Unlike the approaches described above, PRIIME [13] uses *graded feedback*, i.e., it allows the user to rate patterns on the scale from 1 to c, where c is a positive integer greater than 2. (However, in their experiments, the authors only use the fixed value $c = 2$, which is equivalent to binary feedback.) A model that predicts the rating of a pattern is learned from the user feedback by means of multinomial logistic regression. Candidate patterns are assumed to arrive in sequential batches from a stream that is *not* controlled by the learner, e.g., an output of a mining algorithm. Query selection is guided by the potential impact on the model parameters, which is quantified by *expected gradient length*. PRIIME's main strength is advanced feature construction for structured data, which allows it to tackle sequence or graph data.

In this chapter we focus on the *Learn* step of the loop. To implement the *Mine* step, such algorithms as IPM, OCM, and PRIIME use anytime mining techniques (e.g., pattern sampling), which is another issue that we tackle in this thesis. We discuss this aspect in Chapter 5.

Preference learning In this chapter we use preference learning [73], a research area encompassing several tasks related to learning preferences within the field of machine learning. In particular, we deal with an instance of the *object ranking* problem, i.e., acquiring ranking functions from sample orders [78].

Active object ranking is related to the problem of *learning to rank* in information retrieval, as both aim at learning a ranking from a minimum number of sample rankings. A number of general heuristics aimed at improving top results of search engines were developed [125, 152]. Methods that specifically target document ranking algorithms exploit probabilistic models of a document collection [116] or relations between documents [151]. A theoretical analysis of active object ranking shows that it is NP-hard and characterized its query complexity [5].

Simultaneous preference elicitation and learning has been studied in the context of pure optimization problems ("top-1") [23, 32]. In particular, such approaches as *coactive learning* [127] and *constructive preference elicitation* [133, 132] provide theoretical analysis of error bounds and effects of potentially inaccurate feedback. Although the top-1 formulation is too restrictive for pattern mining

(in most cases, there is more than one interesting pattern), these approaches indicate the direction for future extensions of the proposed approach.

3.3 Interactive learning – an example

The following toy example illustrates how the proposed approach can be applied in a data analysis session. Let us assume a subgroup discovery setting and a dataset $\mathcal{D}$ defined over three binary attributes $\mathcal{A} = \{A_1, A_2, A_T\}$, where A_T is the target attribute (see Table 3.1). Furthermore, assume that a user is interested in the relation between A_2 and $A_T = +$, e.g., that a certain property of a loan makes it risky, unless it has other properties. Note that this does not imply that the user knows this a priori, but rather that she would find this pattern interesting once it is shown to her. Hence, asking the user to express this in advance of mining is complicated, but we can hopefully learn this.

	A_1	A_2	A_T
t_1	1	1	+
t_2	1	0	+
t_3	0	1	−

Table 3.1: A toy dataset.

For this small toy example, it is feasible to mine and present all subgroups that occur in the data; see Table 3.2. Moreover, given the above assumption on user interest, we can define a desired, subjective 'target ranking' according to which the patterns should be ranked. That is, there is a ground truth that can be used to emulate user feedback and that we would like to learn. The subgroups are ranked according to this subjective target ranking (the leftmost column).

Rank	Description p	$\lvert C\rvert$	$\lvert C^+\rvert$	$\lvert C^-\rvert$	Sensitivity	Specificity
1	$A_1 = 0 \wedge A_2 = 1$	1	0	1	0	0
2	$A_2 = 1$	2	1	1	0.5	0
3	$A_2 = 0$	1	1	0	0.5	1
4	$A_1 = 1 \wedge A_2 = 0$	1	1	1	0.5	0
5	$A_1 = 0$	1	0	1	0	0
6	$A_1 = 1$	2	2	0	1	1
7	$A_1 = 1 \wedge A_2 = 1$	1	1	1	0.5	0

Table 3.2: All subgroups that occur in the toy dataset of Table 3.1, ranked according to the desired subjective ranking. For each subgroup, its description is given, together with absolute frequencies (all, positive, and negative tuples respectively), sensitivity and specificity.

	Specificity	Subjective rank
$A_1 = 1$	1	6
$A_2 = 0$	1	3
$A_1 = 0$	0	5
$A_2 = 1$	0	2

Table 3.3: Initial subgroups obtained by mining top patterns w.r.t. $Specificity$, limited to patterns consisting of a single condition.

In practice, a user often performs top-k mining with respect to an objective quality measure, and with additional constraints to restrict the results. Here, we assume that the user decides to mine all subgroups consisting of a single condition and rank them according to $Specificity$. The resulting subgroups and ranking is shown in Table 3.3. The obtained ranking does clearly not match our user's desired, subjective ranking. Without our framework, this would be the end result and the user would either have to be happy with the results, or tune the algorithm parameters based on these results. The former is unsatisfactory, while the latter is hard: what other interestingness measure and parameters should we use to obtain more interesting patterns?

The goal of our interactive pattern mining framework is to assist the user in data exploration. To this end, the algorithm proposes the user to inspect and compare small sets of subgroups. In this example, let us assume that the subgroups $\{A_1 = 1; A_2 = 0; A_1 = 0\}$ are selected and shown to the user. The system should provide the user with the tools necessary to inspect and understand the patterns (for example, by means of data visualization); these issues are actively researched in the fields of human-computer interaction and visual analytics.

With the help of these tools, the user gains an understanding of the current patterns, the data, and of her interests, which enables her to provide feedback. In this case, the ranking $A_2 = 0 \succ A_1 = 0 \succ A_1 = 1$ would be given, as this coincides with the subjective, implicit ranking we assume the user to have. This implies that $A_2 = 0$ is deemed subjectively most interesting, while $A_1 = 1$ is the least interesting of the three.

Next, the algorithm uses the obtained feedback to learn a (subjective) pattern ranking function. To be able to use existing learning algorithms for this purpose, patterns are represented as numeric feature vectors (the details are explained in Section 3.5). Then, RankSVM is applied to learn a linear ranking function $h(\vec{p}) = \vec{w} \cdot \vec{p}$, i.e., a vector of pattern feature weights defining a ranking function that is consistent with the user feedback.

Initial	After iteration 1	h	After iteration 2	h	Target
$A_1 = 1$	$A_2 = 0$	-0.01	$A_2 = 1$	-0.01	$A_2 = 1$
$A_2 = 0$	$A_1 = 0$	-0.02	$A_1 = 0$	-0.01	$A_2 = 0$
$A_1 = 0$	$A_2 = 1$	-0.03	$A_2 = 0$	-0.02	$A_1 = 0$
$A_2 = 1$	$A_1 = 1$	-0.06	$A_1 = 1$	-0.1	$A_1 = 1$
$\rho = -0.8$	$\rho = 0.4$		$\rho = 0.8$		

Table 3.4: Learning a desired target ranking from user input. Each iteration of feedback and learning improves the approximation of the target ranking. h denotes the absolute score assigned by the ranking function, ρ is the Spearman's rank correlation coefficient between the indicated and target rankings.

RankSVM views this learning problem as binary classification of difference vectors, which involves minimizing pairwise loss, e.g., the number of discordant pairs. For example, the user feedback above $\{A_2 = 0 \succ A_1 = 0 \succ A_1 = 1\} = \{p_2 \succ p_3 \succ p_1\}$ is translated to a set of pairwise preferences $\{(p_2 \succ p_3), (p_2 \succ p_1), (p_3 \succ p_1)\}$. Given feature representations of patterns $\vec{p_i}$, object ranking can be reduced to positive-only classification of difference vectors, i.e., a ranked pair example $p_i \succ p_j$ corresponds to a classification example $(\vec{p_i} - \vec{p_j}, +)$. All pairs comprise a training dataset for a scoring classifier. Then, the predicted ranking of any set of objects can be obtained by sorting these objects by classifier score descending. (See Section 3.4 for a further discussion of the learning problem and Section 3.5 for a description of pattern features.)

To improve the ranking function, the interaction and learning steps can be repeated a number of times. In our example, the algorithm queries another set of subgroups $\{A_2 = 0; A_2 = 1; A_1 = 1\}$, obtains the ranking $A_2 = 1 \succ A_2 = 0 \succ A_1 = 1$ as feedback, and learns a new weight vector for h. The effect of interaction and learning is shown in Table 3.4. After two iterations, the learned ranking is almost identical to the desired target ranking, and certainly much better than the initial ranking based on specificity. This is also reflected by Spearman's rank correlation coefficients between the obtained and target rankings, indicated by ρ.

This concludes one full iteration of the mine, interact, and learn loop, after which the whole procedure can be repeated. The second loop is executed as the first, except that the learned ranking function is now used to mine and rank patterns.

For this toy example, we evaluate the generalization capacity of the learned ranking function by applying it to the complete set of subgroups from Table 3.2.

Description	h	Learned rank	Target rank
$A1 = 0 \wedge A2 = 1$	0.01	1	1
$A2 = 1$	-0.01	2	2
$A1 = 0$	-0.01	3	5
$A2 = 0$	-0.02	4	3
$A1 = 1 \wedge A2 = 1$	-0.04	5	7
$A1 = 1 \wedge A2 = 0$	-0.04	6	4
$A1 = 1$	-0.1	7	6
			$\rho = 0.8$

Table 3.5: Generalization capacity of the learned ranking function. All subgroups from Table 3.2 are ranked according to h.

The results are presented in Table 3.5. We observe that the learned ranking function generalizes very well: when the subgroups are sorted according to this function, the resulting ranks are very close to those of the desired target ranking. In other words, the learned ranking is highly correlated with the target ranking, indicated by a high correlation coefficient of $\rho = 0.8$. This implies that if h were used as interestingness measure in top-k mining, subjectively more interesting subgroups would be discovered.

In the following sections, we formalize the problem and describe the algorithms required to implement the proposed workflow, i.e., we discuss techniques for learning rankings, query selection methods, and feature representations for patterns.

3.4 Interactive learning of pattern rankings

We now introduce the formal definition of the pattern ranking learning task as well as algorithms for solving this task.

Learning pattern rankings

The pattern ranking task is formally defined as follows. Recall that $\mathcal{L}$ denotes the pattern language, that is, the universal set of all possible patterns. We will assume that there is an unknown, user-specific target ranking R^*, that is, a total order over $\mathcal{L}$. We shall write $p \succ q$ when p is preferred over q according

to R^*. The goal of learning will be to learn an approximation $\hat{R}$ of R^* on the basis of the feedback provided by the user. We make the following assumptions:

The feedback takes the form of example rankings $Q_k^* = p_{1k} \succ \ldots \succ p_{mk}$; these are total strict orders over *subsets* of $\mathcal{L}$. Feedback will be obtained through interaction with the user.

Each hypothesis h (in the hypothesis space $\mathcal{H}$) is a ranking function h that maps descriptions of patterns to real values and defines a ranking as follows: $p_i \succ_h p_j$ if and only if $h(p_i) > h(p_j)$.

Each pattern $p \in \mathcal{L}$ will be represented by a feature vector $\vec{p} = [x_1, \ldots, x_m]$.

The goal is to learn an approximation $\hat{R}$ of the target ranking R^* that minimizes the loss function, on the basis of a set of examples $U = \bigcup Q_k^*$ (a set of example rankings) provided by the user. Note that there is not necessarily a hypothesis $h^* \in \mathcal{H}$ that correctly represents the unknown target ranking R^*. This depends both on the hypothesis space $\mathcal{H}$ considered and the specific target ranking.

The pattern ranking task as just defined can be seen as an instance of object ranking, for which various types of loss functions and learning algorithms have been described. Two principal categories of object ranking techniques are *pairwise* and *listwise* methods.

Pairwise algorithms treat each sample ranking as a set of corresponding ranked pairs. For example, $\vec{p}_1 \succ \vec{p}_2 \succ \vec{p}_3$ corresponds to $\{(\vec{p}_1 \succ \vec{p}_2), (\vec{p}_1 \succ \vec{p}_3), (\vec{p}_2 \succ \vec{p}_3)\}$. In the pairwise setting, the loss that needs to be minimized is a function of the number of incorrectly ranked pairs in the training data:

$$Loss_{pairwise}\left(\hat{R}, U\right) = \sum_{Q_k^* \in U} \sum_{(\vec{p}_{ik} \succ \vec{p}_{jk}) \in Q_k^*} L\left(\hat{R}, \vec{p}_{ik}, \vec{p}_{jk}\right)$$

Listwise algorithms do not reduce the rankings to pairs, i.e., each sample ranking constitutes one training example. Hence, the loss function is defined over complete rankings:

$$Loss_{listwise}\left(\hat{R}, U\right) = \sum_{Q_k^* \in U} L\left(\hat{R}, Q_k^*\right)$$

The pattern ranking task can be solved by finding a ranking $\hat{R}$ that minimizes either a pairwise or a listwise loss function. Since we will use and evaluate instances of both, we will not state a preference here.

Algorithm 1 APLe: Active preference learning for subjective pattern ranking

Input: Dataset $\mathcal{D}$, ranked collection of patterns $\mathcal{P}$
Output: Ranking function h for patterns over $\mathcal{D}$

1: $U = \emptyset$, $h = \mathit{SourceRanking}(\mathcal{P})$, $PV = \textsc{ConvertToVectors}(\mathcal{P}, \mathcal{D})$
2: **repeat**
3: $\quad Q = \textsc{SelectQuery}(PV, h)$
4: $\quad U = U \cup \textsc{GetFeedback}(Q)$
5: $\quad h = \textsc{LearnRankingFunction}(PV, U)$
6: **until** *Stopping criterion* is met
7: **return** h

Algorithm for learning pattern rankings

We now present a generic algorithm for learning pattern ranking functions, dubbed APLe for ***A**ctive **P**reference **Le**arning for subjective pattern ranking* (Algorithm 1). It receives a collection of patterns $\mathcal{P} \subset \mathcal{L}$ as input. The initial pattern collection can be mined using any standard pattern mining algorithm, and is ranked according to an objective interestingness measure. This initial ranking is referred to as the *source ranking*. In order to apply preference learning, patterns are represented as vectors of numeric features (Line 1); this will be discussed in detail in Section 3.5.

Within the interaction and learning loop, query selection methods select sets of patterns that will be shown to the user (Line 3). Assuming that the query size is fixed, the goal is to minimize the number of queries required to attain a certain ranking accuracy. The methods take into account factors such as the current estimated interestingness of a pattern, the estimation uncertainty, the diversity of the query, and/or the structure of the data.

A user provides feedback to the queries in the form of rankings (Line 4). This feedback format is computationally more expensive for a user than graded feedback, i.e., assigning scores from a predefined scale. However, we argue that it has two advantages. First, it requires neither a deep understanding of the scale by a user, nor a thorough scale calibration. Second, graded feedback can be converted to the ordered format, albeit at a cost of reduced granularity.

The ranked queries are then used as training data for an object ranking algorithm. When the pairwise loss function is used, the problem is similar to classification, as illustrated in Section 3.3. In fact, many pairwise algorithms are extensions of classification algorithms, e.g., RankSVM [76], RankBoost [55], and RankNet [29]. Moreover, Stochastic Coordinate Descent (SCD) for logistic loss [124] can be easily adapted to perform pairwise preference learning.

Listwise algorithms are designed specifically for the object ranking task. For example, ListNet [33] uses neural networks and gradient descent to minimize the loss function based on the probabilistic model of permutations. We evaluate the performance of various object ranking algorithms for pattern ranking in Section 3.6.

The interaction and learning loop stops when a certain stopping criterion is met (Line 6). Such criteria can consider marginal effects of additional queries on the learned ranking or limit the maximal user effort. Alternatively, the user can manually stop the algorithm, as soon as she considers her information need satisfied. In the experiments, we stop learning after a fixed number of iterations.

Active learning techniques

Active preference learning is a challenging problem. Selecting an optimal query is NP-hard [5], therefore in most cases exact query selection methods are computationally too expensive to be used in interactive settings. Consequently, heuristic methods are commonly used.

Query selection methods balance exploration of the pattern space with exploitation of available preference feedback. In the context of pattern mining, the source ranking is a strong starting point. The common method to ensure sufficient exploration is to maintain diversity among queried objects. We consider two categories of heuristics: greedy heuristics inspired by methods from information retrieval (IR), which explicitly take objective quality measures into account, and uncertainty-based heuristics specific to the RankSVM learner, as it performed the best in our preliminary experiments (see Table 3.7).

IR-inspired heuristics IR-inspired heuristics were initially developed in the context of improving search engines, hence they inherently aim at identifying a small number of top-ranking objects (*documents*). These greedy heuristics rely on the availability of an objective quality measure (*relevance*). The query selection process always starts from a set including the currently top-ranked pattern and proceeds with greedily selecting patterns that maximize the heuristic. Let p denote a candidate pattern, and Q the current (incomplete) query.

When applying these heuristics, we start from the raw values of the source pattern interestingness measure φ, and progressively interpolate the values of the learned ranking function in order to take into account the current estimation of the target ranking:

$$Quality(p) = \mu \times h(p) + (1 - \mu) \times \varphi(p)$$

where μ is an interpolation parameter and h is the learned ranking function.

MMR (*Maximal Marginal Relevance*) [125] aims to select a high-quality pattern that is dissimilar from already selected patterns. Dissimilarity is defined as the minimal distance to an already selected pattern, e.g., the Euclidean distance between pattern vectors. The parameter $\alpha \in [0; 1]$ is a quality-diversity trade-off parameter.

$$MMR(p, Q) = \alpha\ Quality(p) + (1 - \alpha)\ Diversity(p, Q)$$
$$\text{where } Diversity(p, Q) = \min_{q \in Q} dist(p, q)$$

RDD (*Relevance*, *Diversity*, and *Density*) [152] exploits the structure in $\mathcal{P}$ by adding a density term. The intuition behind this approach is that querying patterns from dense regions provides more information about preferences. Density of a region around a pattern is quantified as the average distance to all other patterns.

$$RDD\,(p, Q) = \alpha\ Quality\,(p) + \beta\ Density\,(p, \mathcal{P}) +$$
$$+(1 - \alpha - \beta)\ Diversity\,(p, Q)$$
$$\text{where } Density\,(p, \mathcal{P}) = \frac{1}{|\mathcal{P}|} \sum_{p' \in \mathcal{P}} dist(p, p')$$

MMR and RDD maintain local diversity, i.e., diversity *within the current query*. We aim to exploit global diversity, i.e., diversity *between queries*, by introducing a new heuristic GlobalMMR. It is an extension of MMR, where the diversity term is redefined as $Diversity(p, \mathcal{Q}) = \min_{p' \in \mathcal{Q}} dist(p, p')$, where $\mathcal{Q} = Q \cup \bigcup_{Q_i^* \in U} Q_i^*$ is the union of all queries, including the current incomplete one.

In all computations, values of the quality measure, the learned ranking function, and the distance measure are normalized to the range $[0; 1]$. For the quality measure and the learned ranking function, the minimal and the maximal values over $\mathcal{P}$ are used as range limits. For distance, the upper limit is estimated by the diameter of the object set, i.e., $\max_{p_i, p_j \in \mathcal{P}} dist(p_i, p_j)$.

Uncertainty-based heuristics SVMBatch, presented in Algorithm 2, is a straightforward extension of the batch query selection method for classification SVMs by Brinker [27]. This method aims at selecting a diverse set of examples with high prediction uncertainty. Uncertainty is quantified as the distance of a candidate example to the margin, whereas diversity is quantified by maximal cosine similarity between an example and already selected examples. This method only considers examples that lie on or within the margins.

Algorithm 2 Batch query selection for RankSVM

Input: Pattern vectors PV, weights w, query size k, trade-off λ, pruning parameter $minlen$

$Q \leftarrow \emptyset$, $Pairs \leftarrow \emptyset$
for all $p_i, p_j \in PV$ **do** ▷ Generate candidate pairs
  $\mathbf{p}_{ij} = p_i - p_j$
  if $dist(\mathbf{p}_{ij}, w) \leq 1 \wedge ||\mathbf{p}_{ij}|| \geq minlen$ **then**
    $Pairs \leftarrow Pairs \cup \mathbf{p}_{ij}$
repeat ▷ Greedily select a diverse set of uncertain pairs
  $\mathbf{p}^*_{ij} = \underset{\mathbf{p} \in Pairs}{\operatorname{argmin}}\ \lambda \times dist(\mathbf{p}, w) + (1 - \lambda) \times \max_{q_i, q_j \in Q} \cos(\mathbf{p}, \mathbf{q}_{ij})$
  $Q \leftarrow Q \cup \{p^*_i, p^*_j\}$
until $|Q| = k$
return Q

In case of pairwise preferences, an individual example is a *pair* of patterns. Pairs are explicitly represented as differences between respective pattern vectors, similar to their representation in the RankSVM formulation. The total number of candidates is proportional to $|\mathcal{P}|^2$, therefore in order to reduce computational costs we introduce an additional pruning step. All pair vectors P_{ij} for which $||P_{ij}|| < minlen$ are removed from the candidate set. The intuition behind this pruning technique is that pair vectors with low norms correspond to highly similar patterns, and reducing uncertainty of predicting relative positions of similar patterns is less useful for learning a general ranking. Note that the distance between a pair vector P_{ij} and the hyperplane is proportional to the difference between values of the ranking function for p_i and p_j. However, the exact value has to be computed explicitly. Recently, Qian et al. [115] proposed a similar active learning heuristic targeted at RankSVM, where efficiency is ensured by combining locality-sensitive and uncertainty hashing.

Preliminary experiments confirmed the utility of batch querying, e.g., querying the union of the two most informative pairs yields a larger performance improvement than three consecutive queries of the single most informative pair (in both cases 6 pairs are queried). Pruning reduces the runtime and can have a positive impact on learning performance; see Section 3.6 for experiments.

Randomized heuristics Initially, we used non-biased uniform sampling of subgroups from $\mathcal{P}$ as a baseline in our experiments. As it resulted in reasonably high learning performance, we decided to further explore randomized query selection methods that sample subgroups proportional to values of IR-inspired heuristics. The overall procedure is as follows:

1. For each pattern, a sampling weight ω is proportional to its score according to a query selection heuristic, e.g. $\omega(p) = \text{MMR}(p, Q)$.
2. If the minimal weight is equal to 0, a Laplace-like correction is applied, i.e., $1/|\mathcal{P}|$ is added to all weights.
3. A random number drawn uniformly from the range $\left[0,\ \sum_{p\in\mathcal{P}} \omega(p)\right]$ determines the sampled pattern.

Hence, the probability of sampling a pattern p is $\omega(p) \Big/ \sum_{p'\in\mathcal{P}} \omega(p')$. To sample a query, k patterns are sampled from $\mathcal{P}$ without replacement.

Mining using learned ranking functions

As also demonstrated in the toy example in Section 3.3, the learned ranking function generalizes beyond the training data U and the input pattern set $\mathcal{P}$. Hence, it can be regarded as a general subjective interestingness measure defined over the pattern language $\mathcal{L}$, for the current user and dataset $\mathcal{D}$. It can be used to discover novel patterns that are likely to be interesting to the user.

A straightforward approach to accomplish this is to use the ranking function h directly as a search heuristic. In addition to h, a search algorithm needs access to the function that converts a pattern to its corresponding feature vector. Given h and the right feature representation, a search algorithm can compute subjective interestingness scores for each pattern $p \in \mathcal{L}$ and hence use this as optimization criterion.

Devising a generic search algorithm for any type of pattern and ranking function h is an interesting and challenging open research problem on itself. We therefore leave this for future work and only specify instances tailored for itemset mining and subgroup discovery in the next section.

3.5 Learning itemset and subgroup rankings

We now describe two instances of our proposed approach, for two types of pattern mining: (frequent) itemset mining and subgroup discovery. The key choices concern the source ranking and the pattern features. In making these choices, we aim to keep things as simple as possible, in order to avoid the necessity of data mining expertise and hence making our approach accessible to domain experts as well.

Frequent itemset mining For itemset mining, in absence of any prior knowledge, the ranking according to frequency is the most natural choice of the source ranking. We consider the following features for itemsets:

- *Attribute* (i): a binary feature for each item; equals 1 iff the corresponding item belongs to the itemset.
- *Cover* (t): a binary feature for each transaction; equals 1 iff the corresponding transaction is covered by the itemset.
- *Frequency*: a numeric feature; the frequency of the itemset.
- *Length*: a numeric feature; the size of the itemset, i.e., $|p|$.

The total number of features depends on the dimensions of the data, and is equal to $|\mathcal{I}| + |\mathcal{D}| + 2$.

Subgroup discovery In case of subgroup discovery, there is more information than just frequency that can be used to start from a –potentially– better source ranking. That is, any objective subgroup interestingness measure φ can be used, e.g., *Sensitivity* or *Specificity*. A standard subgroup discovery algorithm can be used to mine the initial pattern set and corresponding source ranking.

In addition to the features described for frequent itemset mining, which can be used in almost any pattern mining setting, we consider the following features specific to subgroup discovery:

- *Positive Frequency*: a numeric feature; the frequency of the positive labels in the subgroup, i.e., $\left|C_p^+\right| / |\mathcal{D}^+|$.
- *Negative Frequency*: a numeric feature, the frequency of the negative labels in the subgroup, i.e., $\left|C_p^-\right| / |\mathcal{D}^-|$.
- *Quality*: a numeric feature; $\varphi\,(p)$.

Also, due to a richer pattern language, feature sets *Length* and $Attribute_A$ have a slightly different interpretation in case of subgroup discovery: *Length* is equal to the number of conditions in the description, and each attribute is still represented by a single binary feature $Attribute_A$, even if it occurs in multiple conditions. For example, if A_1 and A_2 are numeric attributes, a subgroup $A_1 > 1 \wedge A_1 < 2 \wedge A_2 > 0$ has $Length = 3$, $Attribute\,(A_1) = 1$, and $Attribute\,(A_2) = 1$.

In order to use the learned ranking function h for mining new patterns, we employ a beam search–based subgroup discovery algorithm, DSSD [140]. At each level, h is used to rank candidates in the beam; no other changes to the algorithm are necessary.

3.6 Experiments

In previous sections we described a framework for interactive learning of pattern ranking functions. The key research question is: "Is it possible to learn preferences over patterns, given only sample rankings as input?" We demonstrate that the answer is positive and proceed with answering the following more specific research questions:

Q1 Which ranking algorithms are most suitable for this purpose?

Q2 For which pattern types is learning feasible? If so, how much training data is required?

Q3 Which pattern features are important for learning?

Q4 Does active learning reduce the user effort? Which query selection methods perform better with respect to various performance measures?

Q5 Do the learned ranking functions enable the discovery of novel interesting patterns when used as search heuristics?

Evaluation methodology

User feedback emulation Evaluating interactive data mining algorithms is hard, for experts are scarce, and it is virtually impossible to collect enough data for drawing reliable conclusions. In order to perform an extensive evaluation we use an objective ranking of patterns as the target ranking. We emulate user feedback by ranking patterns using an objective interestingness measure (*target measure*), which is not known to the learning algorithm. We use $Surprisingness$ for itemset mining and χ^2 for subgroup discovery (see Section 2.2 for definitions).

Performance measures Given a set of patterns $\mathcal{P}$, the goal of learning rankings is two-fold: 1) to identify subjectively interesting patterns in $\mathcal{P}$ and 2) to learn an accurate overall ranking of $\mathcal{P}$. Therefore, we use several ranking distance measures to quantify learning performance. Let $R^*_{\mathcal{P}}$ denote the target ranking

of $\mathcal{P}$, $\hat{R}_{\mathcal{P}}$ the learned ranking, and $\hat{R}_{\mathcal{P}}(i)$ the learned rank of the i-th element in the target ranking:

1) In order to evaluate the capacity of the algorithm to identify the most interesting patterns in $\mathcal{P}$, we consider Recall at k:

$$Rec_k = \left| \left\{ i \in \{1, 2, \ldots, k\} \mid \hat{R}_{\mathcal{P}}(i) \leq k \right\} \right|$$

2) In order to evaluate the overall ranking accuracy, we consider rank correlation and discounted error. Spearman's rank correlation coefficient ρ is based on the sum of squared differences between learned and target ranks for each element:

$$\rho = 1 - \frac{6\ D_s(R^*_{\mathcal{P}}, \hat{R}_{\mathcal{P}})}{|\mathcal{P}|(|\mathcal{P}|^2 - 1)}, \text{ where } D_s = \sum (i - \hat{R}_{\mathcal{P}}(i))^2$$

Rank correlation essentially assigns equal weights to all elements, whereas Discounted Error DE assigns larger weights to higher-ranked elements:

$$DE = \sum \frac{|i - \hat{R}_{\mathcal{P}}(i)|}{ln(i+1)}$$

We use ρ as the primary performance measure in the exploratory experiments.

Performance measures calculated for the entire ranking, such as ρ or DE, are less relevant if the ultimate goal is to identify top-ranking patterns. However, if the goal is to learn a search heuristic, the capacity to correctly identify low-ranked patterns is important as well. Note that reported values of DE are normalized to the range of $[0, 1]$.

In order to estimate the convergence rate of the algorithm, for each performance measure we report values of the *area under performance curve* (AUPC) in addition to absolute values. The performance curves are constructed as follows: for each iteration i, the value of a performance measure after i iterations is recorded. The larger the area, the fewer iterations are required to attain high values of the performance measure.

To quantify user effort, we use the total number of distinct queried pairs E_U: $E_U = |\{ (p_{ik}, p_{jk}) \mid Q^*_k \in U; p_{ik}, p_{jk} \in Q^*_k \}|$. E_U is equal to the number of pairwise preferences that a user has to compute in order to provide the feedback.

Datasets and source rankings For our empirical evaluation we used datasets from publicly available repositories: 11 datasets for itemset mining were taken

Setting			**Source rankings**		
Subgroup discovery (SD)			*Frequency*	*Sensitivity*	*Specificity*
Dataset	$\|\mathcal{D}\|$	$\|A\|$	$\rho\,(Source, Target)$		
`breast-w`	683	9	0.26	0.61	0.02
`credit-a`	653	15	−0.26	−0.06	0.51
`credit-g`	1000	20	0.11	0.33	0.86
`diabetes`	768	8	−0.01	0.17	0.43
`vote`	232	16	0.33	0.84	0.51
Itemset mining (FIM)			*Frequency*		
`anneal`	812	94	−0.31		
`australian`	653	125	−0.27		
`german`	1000	112	−0.23		
`heart`	296	95	−0.21		
`hepatitis`	137	68	−0.24		
`lymph`	148	68	0.03		
`primary`	336	31	−0.07		
`soybean`	630	50	0.09		
`tic-tac-toe`	958	27	0.12		
`vote`	435	48	−0.13		
`zoo`	101	36	−0.18		

Table 3.6: Datasets and pattern sets used in experiments. For each dataset, source rankings of 1000 patterns were mined using various objective interestingness measures: *Frequency* for both itemset mining and subgroup discovery, and *Sensitivity* and *Specificity* for subgroup discovery. For each source ranking, Spearman's rank correlation ρ between the source ranking and the target ranking is reported.

from the CP4IM repository[1]; 5 datasets for subgroup discovery were taken from the UCI repository[2]. Tuples with missing attribute values were removed from all datasets.

The 1000 most frequent closed itemsets, ranked by their frequencies, were used as source rankings for experiments with itemset mining. Source subgroup rankings were mined using DSSD with the following parameters (see van Leeuwen and Knobbe [140] for details): minimal frequency = 0.1 $|\mathcal{D}|$, beam width = 100, maximal depth = 5. Numeric attributes were discretized on-the-fly by local

[1] `http://dtai.cs.kuleuven.be/CP4IM/datasets/`
[2] `http://archive.ics.uci.edu/ml/`

Learner	Avg.ρ	Runtime per iteration, s
RankSVM	0.55	0.1
SCD	0.42	0.03
RankBoost	0.38	12.2
ListNet	0.16	2.8
RankNet	0.02	3.5

Table 3.7: Comparison of ranking algorithms. RankSVM provides the best performance in terms of rank correlation ρ and has one of the lowest runtimes.

binning of occurring values into 6 equal-sized bins. The cover-based beam selection heuristic was applied with the default trade-off parameter settings. 10000 subgroups were mined initially, then 1000 subgroups were selected from this large set using the same selection heuristic. For each dataset, three subgroup sets were mined using one of the following subgroup interestingness measures, *Sensitivity*, *Specificity*, or *Frequency* (essentially a non-supervised measure).

We have intentionally chosen simple source measures so that source and target rankings are substantially different, and hence the learning problem is challenging. Table 3.6 presents the characteristics of the datasets and corresponding pattern sets, including the initial rank correlation ρ_0 between the source ranking and the target ranking. Most source rankings by *Frequency* are weakly or negatively correlated with the respective target rankings, whereas source rankings by supervised measures *Sensitivity* and *Specificity* are better correlated with the target rankings by χ^2. In the experiments, we investigate which effect this has on learning performance.

Experimental results

Q1: Comparison of ranking algorithms We first turn to comparing the learning algorithms listed in Section 3.4: ListNet, RankBoost, and RankNet as implemented in the RankLib library[3]; the standard implementation of RankSVM[4]; and our own implementation of SCD.

We use default parameter values in the implementations or values recommended in the original papers: ListNet(1500 epochs, *learning rate* = 0.00001, no hidden layers); RankBoost(300 training rounds, 10 threshold candidates); RankNet(100 epochs, *learning rate* = 0.00005, 1 hidden layer with 10 nodes); RankSVM(trade-off $C = 0.005$) with a linear kernel, per recommendations

[3] http://sourceforge.net/p/lemur/wiki/RankLib/

[4] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

of the authors, it is increased after each iteration, i.e. the effective value is $C_0 \times iteration$; SCD(1000 iterations, regularization parameter = 0.001).

For these experiments, random queries are used as training data. 10 patterns are selected uniformly at random (without replacement) from each source ranking and ranked by the target measure. All algorithms use the same training data. This procedure is repeated 10 times for each source ranking; average values of performance measures are reported. Pattern sets are grouped by the source quality measure, and results are aggregated over all datasets.

Results are shown in Table 3.7. RankSVM, SCD, and RankBoost are able to learn sufficiently accurate rankings, $\rho \gtrsim 0.4$, which indirectly confirms feasibility of our approach. The only listwise algorithm, ListNet, does not perform well, neither does the other neural network-based algorithm RankNet. The results are consistent across pattern types.

We use RankSVM in the following experiments, as it provides the highest performance and has the lowest runtime among the evaluated algorithms. Note that on average, one learning iteration takes approximately $0.1s$, therefore in principle, this implementation can be used in a truly interactive setting.

Q2: Estimating the required amount of training data In order to estimate the amount of required training data, we select uniformly at random S patterns from each source ranking and use them as training data. The average rank correlation over 10 experiments is reported. Figure 3.1 shows the results for $S \in \{0, 10, 30, 50\}$, where $S = 0$ corresponds to the correlation between source and target rankings.

The results show that learning accurate ranking functions requires a reasonable amount of training data: querying at most 30 patterns out of 1000 allows attaining high values of ranking correlation, $\rho \geq 0.7$. They also demonstrate the importance of prior beliefs, i.e., the choice of the source ranking: less training data is required, if the source ranking is better correlated with the target ranking, as is the case for χ^2 and $Sensitivity$ or $Specificity$. Furthermore, the results with the $Frequency$ source ranking are very similar for itemsets and subgroups.

Although these results suggest that preference learning is a suitable technique for ranking patterns, querying 30 patterns at once incurs considerable costs, $E_U = \binom{30}{2} = 435$, which might be prohibitively large for a human user. Later, we demonstrate that active learning helps reduce the required user effort.

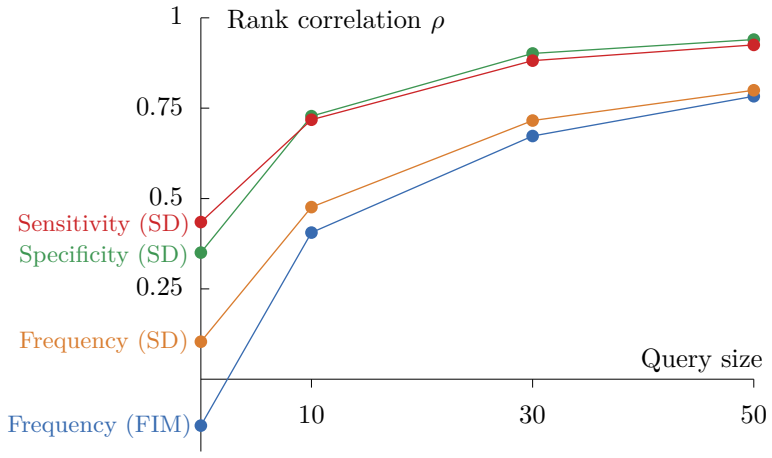


Figure 3.1: Estimating the required amount of training data. Reasonably small training data of 30 ranked patterns or less suffice to attain high values of rank correlation, $\rho \geq 0.7$, Less training data is required, if the source ranking is better correlated with the target ranking, i.e., for the subgroup discovery task and *Sensitivity* and *Specificity* source rankings.

Q3: Evaluating the importance of pattern features In order to evaluate the importance of various feature sets we performed the following procedure. Similar to the previous experiments, random subsets of $\mathcal{P}$ are used as the training data. For each selection of training data, we incrementally construct the pattern representation. At each step the feature set that results in the largest increase of ρ is added to the representation. Note that feature sets such as *Attribute* or *Cover* are added as a whole, as opposed to adding features for each attribute or tuple individually. The procedure continues as long as ρ increases.

For each pattern type, we consider all feature sets described in Section 3.5. Note that all numeric features are discretized into 5 bins. The size of the training data is 30 subgroups. For each subgroup set, the training data selection procedure was performed 10 times; average values are reported.

Results are shown in Table 3.8. The importance of features depends on the pattern type and the target measure. For itemset mining, *Length* was the most likely to be included in the best feature set, because long itemsets tend to have higher values of *Surprisingness*. *Attributes* are important as well, because individual item frequencies are directly included in the formula of *Surprisingness*. For subgroup discovery, features that are included in the

Task	Feature set	First added		In best
		Prob.	ρ	Prob.
SD	Pos.frequency	0.16	0.54	0.75
	Cover	0.65	0.81	0.66
	Neg.frequency	0.15	0.58	0.65
	Quality	0.00	0.32	0.41
	Frequency	0.00	0.59	0.37
	Attributes	0.03	0.47	0.29
	Length	0.00	0.23	0.25
FIM	Length	0.29	0.38	0.90
	Cover	0.55	0.53	0.58
	Attributes	0.16	0.38	0.50
	Frequency	0.00	0.01	0.30

Table 3.8: Evaluating the importance of pattern features. We construct feature representations of patterns incrementally, i.e. we start with an empty representation and add feature sets one by one, based on the improvement of rank correlation ρ that they enable. Features related to the target measures are considerably more likely to be included in the best feature sets, e.g. *Length* for *Surprisingness*, or *Pos./Neg.frequency* for χ^2. For each source ranking, 10 experiments with randomly generated training data were conducted. The first two columns show the probability of a feature set being added at the first iteration and the average attained value of ρ. The rightmost columns show the probability of a feature set being included in the best feature set.

formula of χ^2 are likewise important, for example *Pos./Neg.frequency*. *Cover* is important in both cases, because this feature set helps capture interactions between other features, albeit indirectly. These results also show that the learned weights are interpretable, i.e. that the algorithm can also provide explanations, which may be necessary for human users.

In the remaining experiments, we use the following feature representations: $\{Attributes, Cover, Length\}$ for itemset mining; and $\{Attributes, Cover, PositiveFrequency, NegativeFrequency\}$ for subgroup discovery.

Q4: Query selection We now present the comparison of query selection strategies. We quantify performance by average ranks of strategies with respect to various performance measures. For each source ranking, various query selectors were evaluated and ranked according to AUPC for respective performance measures. Tied ranks are assigned the highest rank from the

equivalent range. Finally, ranks for a specific query selector are averaged over all pattern sets.

Setting parameters First, we briefly describe how to set parameters of query selectors. For IR-inspired selectors MMR and GlobalMMR, we first fix the interpolation coefficient $\mu = 0.5$ and vary the value of α (Table 3.9). The larger focus on query diversity (lower values of α) results in the highest performance; we will use $\alpha = 0.3$ in experiments. For RDD, we essentially keep the same weight assigned to the diversity component (0.7) and vary the values of α and β so that $\alpha + \beta = 0.3$ (for completeness, we also provide results for two combinations with a lower diversity weight). The performance is slightly better than that of MMR and does not differ substantially for various combinations of α and β; we will use $\alpha = 0.15, \beta = 0.15$ in experiments. Finally, for the chosen parameter values, we vary the value of μ. The effect on performance is small; we will use $\mu = 0.5$ in experiments. We always use the Euclidean distance measure.

α	**Avg.rank**	**Avg.ρ**
0.3	2.1	0.52
0.1	2.7	0.52
0.5	3.1	0.41
0.9	3.4	0.37
0.7	3.6	0.35

a) MMR

α	β	**Avg.rank**	**Avg.ρ**
0.15	0.15	2.6	0.54
0.1	0.2	2.7	0.55
0.2	0.1	2.7	0.53
0.25	0.25	3.1	0.47
0.45	0.45	3.8	0.36

b) RDD

α	**Avg.rank**	**Avg.ρ**
0.3	1.8	0.67
0.1	1.9	0.68
0.5	2.9	0.57
0.7	3.9	0.49
0.9	4.5	0.42

c) GlobalMMR

μ	**Avg.rank**	**Avg.ρ**
0.5	2.3	0.58
0.7	2.6	0.56
0.3	2.7	0.55
0.9	2.9	0.51
0.1	3.0	0.52

d) Interpolation

Table 3.9: Tuning IR-inspired active learning heuristics. For each source ranking, we rank parameter values according to attained values of rank correlation ρ and report average ranks across all source rankings. Lower values of α and β, i.e., increasing diversity of selected queries, improves the performance of IR-inspired selectors. We include MMR(0.3), RDD(0.15, 0.15), and GlobalMMR(0.3) in our experiments. The effect of the interpolation coefficient μ is not substantial; we use $\mu = 0.5$ in experiments.

λ	**Avg.rank**	**Avg.ρ**
0.3	2.8	0.69
0.7	2.8	0.68
0.5	3.0	0.66
0.1	3.1	0.69
0.9	3.2	0.66

minlen	**Avg.rank**	**Avg.ρ**
0.3	2.0	0.71
–	2.7	0.69
0.1	3.1	0.68

Table 3.10: Tuning an uncertainty-based active learning heuristic SVMBATCH. Performance of SVMBATCH does not depend substantially on the uncertainty weight λ. Pruning a candidate set can improve performance. We use $\lambda = 0.3$ and $minlen = 0.3$ in experiments.

For SVMBATCH, we first turn off candidate set pruning and vary the values of the uncertainty weight λ (Table 3.10). In line with original findings [27], the effect on performance is small; we will use $\lambda = 0.3$ in experiments. Then, for the chosen λ, we experiment with values of the pruning threshold *minlen*, where $minlen = x$ denotes pruning all candidate pairs with the norm less than $x \cdot \sqrt{2d}$ and $\sqrt{2d}$ is the maximal norm of a binary vector of the dimensionality d (the dimensionality of a pattern feature vector depends on the dimensions of the dataset and the chosen feature sets). We observe that pruning can potentially improve the performance, hence we will use $minlen = 0.3$ in experiments. Note that larger values of *minlen* in certain cases can result in overly eager pruning and hence in empty candidate sets; therefore they are not reported in the table.

Comparison of query selection heuristics Following the results of previous experiments, we compare the following heuristics: IR-inspired selectors MMR($\alpha = 0.3$), RDD($\alpha = 0.1$, $\beta = 0.2$), and GLOBALMMR($\alpha = 0.3$) with $\mu = 0.5$ and the Euclidean distance measure; SVMBATCH($\lambda = 0.1$, $minlen = 0.1$). A non-biased randomized strategy RANDOM, which selects subsets of the source ranking uniformly at random, is used as a baseline. To compute the ranks of RANDOM, for each experimental setting, 10 experiments were conducted, and median values of performance measures were used.

All experiments were conducted with 10 iterations and query size $S = 5$. The maximal effort is then $E_U = 10 \times \binom{5}{2} = 100$. A single query of 15 patterns has roughly equivalent costs, $E_U = \binom{15}{2} = 105$, therefore we report the median performance over 10 experiments with RANDOM and $S = 15$ as a non-iterative baseline.

Table 3.11 presents the aggregate results regarding the performance of query selectors. They show that global query diversity is required to learn accurate

Selector	Left column: average rank. Right column: average value after 10 iterations. ρ		DE		Rec_{10}		Rec_{100}		E_U	
SVMBatch	2.2	0.73	2.2	0.24	1.5	0.4	2.2	0.67	3.3	99.9
GlobalMMR	2.3	0.73	2.2	0.24	1.3	0.4	2.2	0.69	3.1	94.2
Random	3.0	0.74	3.3	0.26	3.0	0.2	3.8	0.58	3.2	100
RDD	3.5	0.64	3.4	0.32	2.2	0.3	3.2	0.56	1.6	46.2
MMR	3.7	0.63	3.5	0.32	2.3	0.3	3.1	0.58	1.4	44.1
Random(S=15)		0.64		0.32		0.1		0.48		105

Table 3.11: Comparison of active learning heuristics. For each source ranking, heuristics are ranked based on the values of respective performance measures; we report the average ranks across all source rankings and average values of performance measures after 10 iterations. Performance of all heuristics is comparable to the non-iterative baseline. Methods that ensure global diversity, GlobalMMR and SVMBatch, result in accurate overall rankings, i.e. rank highly according to rank correlation ρ and discounted error DE. MMR and RDD provide slightly lower performance, but at considerably lower costs E_U. Iterative random query selection performs well in terms of learning overall rankings (ρ), but is outperformed in terms of recall at the top of the ranking (Rec_{10} and Rec_{100}).

overall rankings: methods that ensure global diversity, i.e., GlobalMMR, SVMBatch, and Random, attain the highest values of ρ and DE. However, active learning heuristics slightly outperform Random in terms of DE, i.e., they are more accurate at the top of the ranking. The performance of IR-inspired selectors, MMR and RDD, is substantially lower, but acceptable, i.e., it is comparable to the baseline. However, they incur considerably lower costs: they query approximately two times fewer pattern pairs. Also, their recall at the top of the ranking is substantially larger than for the random query selection.

Table 3.12 shows results grouped by source measures. The performance of active learning strongly depends on the source ranking. For the source rankings highly correlated with the target ranking, i.e., the subgroup discovery task and the *Sensitivity* and *Specificity* source rankings, active learning heuristics outperform random query selection according to most performance measures.

Randomized query selection Table 3.13 compares the IR-inspired heuristics MMR and RDD with their randomized variants as well as the uniformly random query selection and SVMBatch in the subgroup discovery task. Sampling essentially emphasizes global diversity, thus, as expected, the randomized

Task	Source	Selector	ρ		Rec_{10}	
			Left column: avg.rank.	Right column: avg.value after 10 iterations.		
SD	Frequency	RANDOM	1.5	0.71	3.8	0.2
		GLOBALMMR	2.2	0.59	1.4	0.6
		SVMBATCH	2.8	0.46	2.4	0.5
		MMR	4.0	0.38	3.2	0.3
		RDD	4.4	0.40	3.2	0.2
	Sensitivity	SVMBATCH	1.2	0.94	1.6	0.8
		GLOBALMMR	2.0	0.95	2.0	0.8
		RDD	3.6	0.85	2.8	0.6
		MMR	3.6	0.89	3.4	0.6
		RANDOM	4.0	0.87	4.3	0.2
	Specificity	SVMBATCH	1.2	0.93	1.8	0.8
		GLOBALMMR	2.2	0.91	1.4	0.9
		RDD	3.4	0.86	3.2	0.7
		RANDOM	3.7	0.85	4.3	0.5
		MMR	4.0	0.86	3.0	0.7
FIM	Frequency	GLOBALMMR	2.5	0.62	2.5	0.2
		SVMBATCH	2.9	0.66	2.9	0.2
		RANDOM	3.0	0.62	3.0	0.1
		RDD	3.2	0.55	3.2	0.1
		MMR	3.5	0.52	3.5	0.1

Table 3.12: Comparison of active learning heuristics; results are grouped by source measures. Performance of query selectors depends on the source ranking. Active learning outperforms iterative random query selection both with respect to overall ranking correlation ρ and recall at the top of the ranking Rec_{10}, when the source ranking is correlated with the target ranking, i.e. for the subgroup discovery task and the *Sensitivity* or *Specificity* source rankings.

heuristics perform well in terms of overall ranking accuracy ρ at the expense of recall at the top Rec_{10} and higher effort E_U. Owing to the bias in sampling, the average target rank in a query is higher than for the purely random selection. Furthermore, their overall performance is comparable to SVMBATCH, which is tailored for the RANKSVM learner. This shows the potential of randomized query selection that is biased towards higher-quality patterns.

Q5: Generalizing to the entire pattern language Finally, we evaluate the capacity of learned ranking functions to generalize to unobserved patterns: we estimate the target interestingness of top-k patterns according to learned

Method	Randomized?	Rec_{10}	ρ	E_U	Avg.target rank in a query
MMR		0.54	0.68	60.6	186.5
	✓	0.34	0.82	99.9	226.1
RDD		0.47	0.70	58.2	227.6
	✓	0.35	0.81	99.9	231.8
SVMBatch		0.43	0.80	99.9	228.4
Random	✓	0.32	0.81	100.0	239.4

Table 3.13: Randomized IR-inspired heuristics emphasize global diversity and thus improve the overall ranking accuracy as measured by ρ at the expense of recall at the top Rec_{10} and effort E_U. Their performance is comparable to that of SVMBatch, which is tailored for the RankSVM learner.

ranking functions, which do not necessarily belong to the source ranking $\mathcal{P}$, and compare it with the interestingness of patterns that are obtained by the search guided by the target measures directly. We use $k = 1000$.

For itemset mining, we first mine a complete collection of frequent itemsets at $\sigma = 0.1$ and rank it using *Surprisingness* and the learned ranking function h to obtain the top-k patterns. We restrict ourselves to the datasets that contain less than 1 million itemsets at this support threshold: *primary-tumor* (50040 itemsets), *soybean* (27635 itemsets), *tic-tac-toe* (1661 itemsets), *vote* (49097 itemsets), and *zoo-1* (151806 itemsets). For subgroup discovery, we use DSSD to search with χ^2 and its extension as described in Section 3.5 to search with the learned ranking functions. Search parameters were identical to the parameters used for mining the source rankings, and learning parameters were identical to the ones used in the query selection experiments.

For each dataset, we learn a ranking function h for a number of iterations and use it to mine novel subgroups or rank a complete collection of frequent itemsets. $\Delta\varphi_{med}$ (resp. $\Delta\varphi^*_{med}$) denotes the ratio between the median values of the target measure φ in top 1000 patterns according to h (mined or ranked) and in the source ranking (resp. in top 1000 subgroups according to φ directly), whereas $\Delta\varphi_{max}$ and $\Delta\varphi^*_{max}$ denote the ratios between the maximal values of φ in respective sets. For each selector, we report the median values of these ratios across all datasets and source rankings.

The results confirm the generalization capacity of learned ranking functions (Table 3.14): median values of the target measures of top k patterns according to h increase substantially, when compared to source rankings. Maximal values

Task	Selector	Iterations	Avg.ρ	$\Delta\varphi_{med}$	$\Delta\varphi_{max}$	$\Delta\varphi^*_{med}$	$\Delta\varphi^*_{max}$
SD	GlobalMMR	5	0.66	1.85	1.03	0.54	0.96
		10	0.74	2.05	1.03	0.64	0.96
	MMR	5	0.60	1.73	1.02	0.34	0.94
		10	0.58	1.70	1.01	0.27	0.92
	SVMBatch	5	0.65	2.63	1.02	0.43	0.98
		10	0.76	3.29	1.02	0.68	0.96
FIM	GlobalMMR	5	0.47	2.09	0.84	0.57	0.83
		10	0.61	2.16	0.91	0.58	0.89
	MMR	5	0.35	2.22	0.87	0.65	0.84
		10	0.38	2.30	0.87	0.58	0.84
	SVMBatch	5	0.41	3.38	0.93	0.68	0.89
		10	0.63	2.74	0.84	0.54	0.81

Table 3.14: Evaluating generalization capacity of learned ranking functions. Learned ranking functions generalize beyond the source rankings, as evidenced by the increase of median target measure values ($\Delta\varphi_{med} > 1$). Learning accurate overall rankings (higher values of ρ) improves quality of discovered patterns; the effect is more evident for subgroup discovery than for itemset mining.

are comparable to what can be achieved with direct search. Moreover, learning accurate rankings increases the magnitude of improvement. For this reason, GlobalMMR or SVMBatch result in better generalization than MMR.

The generalization performance is lower in the case of itemset mining, due to a source ranking that is less correlated with the target. This makes overfitting more likely; in other words, the learned ranking functions are only applicable to $\mathcal{P}$, but not to the entire $\mathcal{L}$. This is the case for SVMBatch: larger values of ρ result in lower *Surprisingness* of top-ranked itemsets.

Figure 3.2 presents a detailed view of two experiments with SVMBatch, with the dataset *primary tumor* for itemset mining and the dataset *credit-a* and the source ranking by *Specificity*. The ranking functions learned after 1, 2, 5, and 10 iterations were used in the search. The boxplots show the distribution of the target measures (*Surprisingness* and χ^2 respectively) in the set of top-1000 patterns according to the learned ranking function. They illustrate the phenomena discussed in the previous paragraph. For itemset mining, overfitting results in decrease of the maximal *Surprisingness* after more learning iterations.

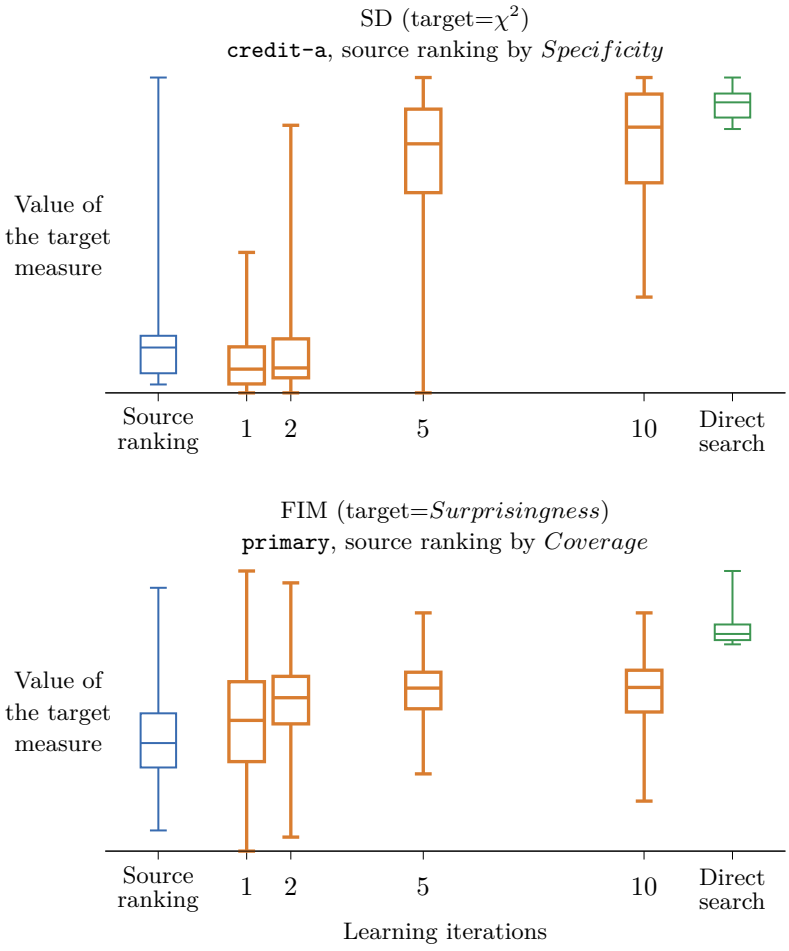


Figure 3.2: Generalization capacity of learned ranking functions. We use a ranking function learned after a certain number of iterations to mine or rank complete collections of patterns. The more learning iterations are performed, the higher the values of the target measure of the patterns discovered with the learned ranking function as a search heuristic. For subgroup discovery, the results after 10 learning iterations are comparable to the search directly guided by the target measure χ^2. For itemset mining, the learning is more prone to overfitting, therefore the maximal value of the target of discovered itemsets slightly decreases. Nevertheless, the median gradually increases.

Nevertheless, the median gradually increases. For subgroup discovery, the more learning iterations are performed, the more the distributions are skewed towards high values of χ^2. Median and maximal values are comparable to ones obtained with χ^2 used directly as a search heuristic.

3.7 Discussion

We introduced a generic algorithm for the interactive learning of pattern rankings, based on off-the-shelf preference learning techniques and active learning heuristics adapted from information retrieval and classification. Furthermore, we presented two instances of this algorithm for well-known pattern mining settings, namely subgroup discovery and itemset mining. Design choices that are specific to each setting include the feature representation of patterns and the choice of source rankings, which represent the prior beliefs of a user. We investigated straightforward and simple options for both of these, by using basic features that follow directly from the problem statement and standard objective measures to define source rankings. Nevertheless, experiments confirm that the proposed algorithm has the capacity to learn accurate pattern rankings in both settings. Moreover, the learned ranking functions generalize beyond the source rankings and hence can be used to mine novel patterns.

These results imply that active preference learning can become an important building block for interactive pattern mining systems, which allow a user to directly influence the mining process so that the results are more relevant to her interests and goals. Such systems should be transparent to non-data mining experts and be able to learn from easy-to-provide feedback. To this end, requiring strict total orders as feedback on complete queries is relatively complicated. Binary feedback, e.g., *liking* or *disliking* patterns, is more intuitive for users. In fact, pairwise ranking algorithms, such as RankSVM, do not require total orders as input and are directly applicable to any feedback format that can be converted to pairwise preferences. Therefore, designing simpler feedback formats, e.g., implicit feedback that is inferred from user actions, and investigating the effects of coarse-grained feedback on the performance are important future directions.

Source rankings were shown to have a considerable effect on the performance of the learning algorithm. Although this is to be expected, this also introduces a non-trivial parameter for non-expert users. Moreover, if a source ranking does not contain information relevant to the target preferences, the learning algorithm is more prone to overfitting and learned ranking functions do not generalize to the entire pattern language $\mathcal{L}$. One way to alleviate this issue

is to move from query selection to *query synthesis*, i.e., mining novel patterns for querying instead of selecting them from a pre-mined pool. This would produce more representative queries and takes elicited preferences into account more rapidly. Pattern sampling can be used to achieve these goals without the overhead of exhaustive mining in each iteration. To this end, we investigate pattern sampling in detail in Chapters 4 and 5.

Finally, to evaluate our algorithm, we emulated the subjective rankings with rankings according to a (latent) objective interestingness measure. These target rankings are total orders, therefore they belong to the hypothesis space $\mathcal{H}$. Whether this assumption holds in practice, i.e., whether genuine subjective pattern rankings can be modeled with total orders, is an open question. Real-world case studies are required to validate the proposed approach.

3.8 Conclusions

We presented a general framework for interactive learning of pattern rankings. It requires a user to rank sets of patterns by perceived interestingness and uses preference learning to infer a general ranking function from these sample rankings. An active learning component is used to minimize user effort. The learned ranking functions generalize well and can be used as a search heuristic, enabling the discovery of novel, potentially more interesting patterns.

We applied this framework to two types of pattern mining: frequent itemset mining and subgroup discovery, which can be considered examples of unsupervised and supervised pattern mining respectively. Using a well-principled evaluation method based on user emulation, we demonstrated that it is possible to learn complex preferences over sets of patterns using off-the-shelf preference learning algorithms. Experiments with active learning heuristics showed a trade-off between accuracy of learned rankings and user effort.

In addition to query synthesis, which we study in the following chapters, possible directions for future work include investigating the effect of coarse-grained or noisy feedback on learning performance and a user study to evaluate the practical applicability of the proposed framework.

Chapter 4

Flexible pattern sampling with guarantees

4.1 Introduction

Traditional pattern mining methods enumerate all frequent patterns, but it is well-known that this usually results in humongous amounts of patterns, i.e., the pattern explosion. To make pattern mining more useful for exploratory purposes, different solutions to this problem have been proposed. In Section 2.2 we listed a number of these solutions along with their advantages and disadvantages. In brief, *condensed representations* [31] can often be efficiently mined, but generally still result in large numbers of patterns. *Top-k mining* [154] is efficient but results in strongly related, redundant patterns showing a lack of diversity. *Constrained mining* [107] may result in too few or too many patterns, depending on the user-chosen constraints. *Pattern set mining* [25] takes into account the relationships between the patterns, which can result in small solution sets, but is computationally intensive.

In this chapter, we study *pattern sampling*, another approach that has been proposed recently: instead of enumerating all patterns, patterns are sampled one by one, according to a probability distribution that is proportional to a

This chapter is based on the journal article "Flexible constrained sampling with guarantees for pattern mining" [48].

given quality measure. The promised benefits include: 1) flexibility in that potentially a broad range of quality measures and constraints can be used; 2) 'anytime' data exploration, where a growing representative set of patterns can be generated and inspected at any time; 3) diversity in that the generated sets of patterns are independently sampled from different regions in the solution space. To be reliable, pattern samplers should provide theoretical guarantees regarding the sampling accuracy, i.e., the difference between the empirical probability of sampling a pattern and the target probability determined by its quality. These properties are essential for pattern mining applications ranging from showing patterns directly to the user, where flexibility and the anytime property enable experimenting with and fine-tuning mining task formulations, to candidate generation for building pattern-based models, for which the approximation guarantees can be derived from those of the sampler.

The performance of the randomized active learning heuristics in the experiments in Chapter 3 provides additional motivation. Recall that these heuristics sample patterns from a fixed pool that needs to be mined *before* the interactive session. Moreover, the choice of this pool, which we referred to as the source ranking, has a strong influence on learning performance. In contrast, a pattern sampler samples patterns from the entire pattern language (subject to constraints) in an anytime manner, i.e., on demand. Thus in the context of interactive mining, another promised benefit of pattern sampling is the transition from pool-based active learning to *query synthesis*, which increases the flexibility of the system and potentially improves its performance.

While a number of pattern sampling approaches have been developed over the past years, they are either inflexible (as they only support a limited number of quality measures and constraints), or do not provide theoretical guarantees concerning the sampling accuracy. At the algorithmic level, they follow standard sampling approaches such as Markov Chain Monte Carlo random walks over the pattern lattice [15, 72, 14], or a special purpose sampling procedure tailored for a restricted set of itemset mining tasks [17, 19]. Although MCMC approaches are in principle applicable to a broad range of tasks, they often converge only slowly to the desired target distribution and require the selection of the "right" proposal distributions.

To the best of our knowledge, none of the existing approaches to pattern sampling takes advantage of the latest developments in sampling technology from the SAT-solving community, where a number of powerful samplers based on random hash functions and XOR-sampling have been developed [66, 37, 51, 101]. WeightGen [36], one of the recent approaches developed for SAT-sampling, possesses the benefits mentioned above: it is an anytime algorithm, it is flexible as it works with any distribution, it generates diverse solutions, and provides strong performance guarantees under reasonable assumptions.

Sampler	Arbitrary constraints	Arbitrary distributions	Strong guarantees	Efficiency	Pattern set sampling
ACFI [15]	Minimal frequency	-	-	✓	-
LRW [72]	✓	✓	-	Implementation-specific	-
FCA [14]	Anti-/ monotonic	✓	-	✓	-
TS (Two-step) [17, 19]	-	-	✓	✓	-
FLEXICS	GFLEXICS	✓	✓	EFLEXICS	✓

Table 4.1: FLEXICS, the method described in this chapter, is the first pattern sampler that combines flexibility with respect to the choice of constraints and sampling distributions with strong theoretical guarantees.

In this chapter, we show that the latest developments in SAT sampling are also relevant to pattern sampling and essentially offer the same advantages. Our results build upon the view of pattern mining as constraint satisfaction, which is now commonly accepted in the data mining community [68].

Approach and contributions More specifically, we introduce FLEXICS: a flexible pattern sampler that samples from distributions induced by a variety of pattern quality measures and allows for a broad range of constraints while still providing strong theoretical guarantees. Notably, FLEXICS is, in principle, agnostic of the quality measure, as the sampler treats it as a black box. (However, its properties affect the efficiency of the algorithm.) The other building block is a *constraint oracle* that enumerates all patterns that satisfy the constraints, i.e., a mining algorithm. The proposed approach allows converting an existing pattern mining algorithm into a sampler with guarantees. Thus, its flexibility is not limited by the choice of constraints and quality measures, but even allows tackling richer pattern languages, which we demonstrate by tackling the novel task of *sampling sets of patterns*. Table 4.1 compares the proposed approach to alternative samplers; see Section 4.3 for a more detailed discussion.

The main technical contribution consists of two variants of the FLEXICS sampler, which are based on different constraint oracles. First, we introduce a generic variant, dubbed GFLEXICS, that supports a wide range of pattern constraints,

such as syntactic or redundancy-eliminating constraints. GFlexics uses cp4im [68], a declarative constraint programming-based mining system, as its oracle. Any constraint supported by cp4im can be used without interfering with the umbrella procedure that performs the actual sampling task. Unlike the original version of WeightGen that is geared towards SAT, GFlexics can handle cardinality constraints, which are ubiquitous in pattern mining. Furthermore, we identify (based on previous research) the properties of the constraint satisfaction-based formalization of pattern mining that further improve the efficiency of the sampling procedure without affecting its guarantees and thus make it applicable to practical problems. We use GFlexics to tackle a wide range of well-known itemset sampling tasks as well as the novel pattern set sampling task. Second, as it is well-known that generic solvers impose an overhead on runtime, we introduce a variant specialized towards frequent itemsets, dubbed EFlexics, which has an extended version of Eclat [153] at its core as oracle.

Experiments show that Flexics' sampling accuracy is impressively high: in a variety of settings supported by the sampler, empirical frequencies are within a small factor of the target distribution induced by various quality measures. Furthermore, practical accuracy is substantially higher than theory guarantees. EFlexics is shown to be faster than its generic cousin, demonstrating that developing specialized solvers for specific tasks is beneficial when runtime is an issue. Finally, the flexibility of the sampler allows us to use the same approach to successfully tackle the novel problem of sampling pattern sets. This demonstrates that Flexics is a useful tool for pattern-based data exploration.

This chapter is organized as follows. We formally define the problem of pattern sampling in Section 4.2. After reviewing related research in Section 4.3, we present the two key ingredients of the proposed approach in Section 4.4: 1) the perspective on pattern mining as a constraint satisfaction problem and 2) hashing-based sampling with WeightGen. In Section 4.5, we present Flexics, a flexible pattern sampler with guarantees. In particular, we outline the modifications required to adapt WeightGen to pattern sampling and describe the procedure to convert two existing mining algorithms into oracles suitable for use with WeightGen, which yields two variants of Flexics. In Section 4.7, we introduce the pattern set sampling task and describe how it can be tackled with Flexics. We also outline sampling non-overlapping tilings, an example of pattern set sampling that is studied in the experiments. The experimental evaluation in Section 4.8 investigates the accuracy, scalability, and flexibility of the proposed sampler. We discuss its potential applications, advantages, and limitations in Section 4.9. Finally, we present our conclusions in Section 4.10.

4.2 Problem definition

Here we present a high-level definition of the task that we consider in this chapter; for concrete instances and examples, see Sections 4.4 and 4.7. The pattern sampling problem is formally defined as follows: given a dataset $\mathcal{D}$, a pattern language $\mathcal{L}$, a set of constraints $\mathcal{C}$, and a quality measure $\varphi : \mathcal{L} \to \mathbb{R}^+$, generate random patterns that satisfy constraints in $\mathcal{C}$ with probability proportional to their qualities:

$$P_\varphi(p) = \begin{cases} \varphi(p)/Z_\varphi & \text{if } p \in \mathcal{L} \text{ satisfies } \mathcal{C} \\ 0 & \text{otherwise} \end{cases}$$

where Z_φ is an (often unknown) normalization constant.

A quality measure quantifies the domain-specific interestingness of a pattern. The choice of a quality measure and constraints allows a user to express her analysis requirements. The sampling procedure meets these requirements by satisfying the constraints and generating high-quality patterns more frequently. Thus, sampled patterns are a representative subset of all interesting regularities in the dataset.

Pattern set mining is an extension of pattern mining, which considers sets of patterns rather than individual patterns. Despite its popularity, we are not aware of the existence of pattern set samplers. The task of pattern set sampling can easily be formalized as an extension of pattern sampling, where we sample sets of patterns $s \subset \mathcal{L}$, and the constraints $\mathcal{C}$ as well as the quality measure φ are specified over sets of patterns (from $2^{\mathcal{L}}$) rather than individual patterns (from $\mathcal{L}$).

4.3 Related work

We here focus on two classes of related work, i.e., 1) pattern mining as constraint satisfaction and 2) pattern sampling.

Constrained pattern mining The study of constraints has been a prominent subfield of pattern mining. A wide range of constraint classes were investigated, including anti-monotonic constraints [3], convertible constraints [112], and others. Another development of these ideas led to the introduction of global constraints that concern multiple patterns and to the emergence of *pattern set*

mining [86, 41]. Furthermore, generic mining systems that could freely combine various constraints were proposed [28, 20].

These insights allowed to draw a connection between pattern mining and constraint satisfaction in AI, e.g., SAT or constraint programming (CP). As a result, declarative mining systems, which use generic constraint solvers to mine patterns according to a declarative specification of the mining task, were proposed. For example, CP was used to develop first declarative systems for itemset mining [68] and pattern set mining [80, 69]. Recently, declarative approaches have been extended to support sequence mining [79] and graph mining [109].

Constraint-based systems allow a user to specify a wide range of pattern constraints and thus provide tools to alleviate the pattern explosion. However, the underlying solvers use systematic search, which affects the order of pattern generation and thus prevents them from being used in a truly anytime manner due to low diversity of consecutive solutions. Similarly, pattern set miners that directly aim at obtaining diverse result sets typically incur prohibitive computational costs as the size of the pattern space grows.

Pattern sampling Here we focus on the approaches that directly aim at generating random pattern collections rather than the methods whose goal is to estimate dataset or pattern language statistics; cf. Shervashidze et al. [126].

Table 4.1 compares our method with the approaches described in Section 4.1, namely MCMC and *two-step* samplers [17, 19]. We further break down MCMC samplers into three groups: ACFI, the very first uniform sampler developed for approximate counting of frequent itemsets [15]; LRW, a generic approach based on random walks over pattern lattice [72]; and FCA, a sampler, which uses Markov chains based on insights from formal concept analysis [14].

Although MCMC samplers provide theoretical guarantees, in practice, their convergence is often slow and hard to diagnose. Solutions such as long burn-in or heuristic adaptations either increase the runtime or weaken the guarantees. Furthermore, ACFI is tailored for a single task; FCA only supports anti-/monotone constraints; and LRW checks constraints locally, while building the neighborhood of a state, which might require advanced reasoning and extensive caching. Two-step samplers, while provably accurate and efficient, only support a limited number of weight functions and do not support constraints.

Constraint	Parameters	CP formulation
$coverage$		$\forall t \in \mathcal{T}\ T_t = 1 \Leftrightarrow \sum_{i \in \mathcal{I}} I_i \left(1 - \mathcal{D}_{ti}\right) = 0$
$minfreq\left(\theta\right)$	$\theta \in (0,\ 1]$	$\forall i \in \mathcal{I}\ \ I_i = 1\ \Rightarrow \sum_{t \in \mathcal{T}} T_t \mathcal{D}_{ti} \geq \theta \times \lvert\mathcal{D}\rvert$
$closed$		$\forall i \in \mathcal{I}\ \ I_i = 1\ \Leftrightarrow \sum_{t \in \mathcal{T}} T_t \left(1 - \mathcal{D}_{ti}\right) = 0$
$minlen\left(\lambda\right)$	$\lambda \in [1,\ \lvert\mathcal{I}\rvert]$	$\forall t \in \mathcal{T}\ T_t = 1\ \Rightarrow \sum_{i \in \mathcal{I}} I_i \mathcal{D}_{ti} \geq \lambda$

Table 4.2: Constraint programming formulations of common itemset mining constraints. $I_i = 1$ implies that item i is included in the current (partial) solution, whereas $T_t = 1$ implies that it covers transaction t.

4.4 Preliminaries

We first formalize itemset mining as a constraint satisfaction problem (CSP) and then describe WeightGen, a hashing-based sampling algorithm.

Itemset mining as constraint satisfaction

We first give a brief overview of the general approach to solving CSPs and then present a formalization of itemset mining as a CSP, following that of cp4im [68]. Formally, a CSP is comprised of *variables* along with their *domains* and *constraints* over these variables. The goal is to find a solution, i.e., an assignment of values to all variables that satisfies all constraints. Every constraint is implemented by a *propagator*, i.e., an algorithm that takes domains as input and removes values that do not satisfy the constraint. Propagators are activated when variable domains change, e.g., by the search mechanism or other propagators. A CSP solver is typically based on depth-first search. After a variable is assigned a value, propagators are run until domains cannot be reduced any further. At this point, three cases are possible: 1) a variable has an empty domain, i.e., the current search branch has failed and backtracking is necessary, 2) there are unassigned variables, i.e., further branching is necessary, or 3) all variables are assigned a value, i.e., a solution is found.

Recall the definition of itemset mining in Section 2.1: $\mathcal{D}$ denotes a dataset, $\mathcal{I}$ denotes the set of items, and $\mathcal{T}$ denotes the set of transaction indices. Let I_i denote a variable corresponding to each item; T_t a variable corresponding to each transaction; and $\mathcal{D}_{ti}$ a constant that is equal to 1, if item i occurs in transaction t, and 0 otherwise. Variables I_i and T_t are *binary*, i.e., their domain is $\{0, 1\}$. Each CSP solution corresponds to a single itemset. Thus, for example, $I_i = 1$ implies that item i is included in the current (partial) solution, whereas $T_t = 0$ implies that transaction t is *not* covered by it. Table 4.2 lists some of

the most common constraints. The *coverage* constraint essentially models a dataset query and ensures that if the item variable assignment corresponds to an itemset p, only those transaction variables that correspond to indices of transactions where p occurs, are assigned value 1. Other constraints allow users to remove uninteresting solutions, e.g., redundant non-*closed* itemsets. Most solvers provide facilities for enumerating all solutions in sequence, i.e., to enumerate all patterns.

WeightGen

WeightGen [36] is an algorithm for approximate weighted sampling of satisfying assignments (solutions) of a Boolean formula that only requires access to an efficient constraint oracle that enumerates the solutions, e.g., a SAT solver. The core idea consists in partitioning the solution space into a number of random subsets, referred to as "cells", and sampling a solution from a random cell. Partitioning with desired properties is obtained via augmenting the original problem with random XOR constraints. Theoretical guarantees stem from the properties of uniformly random XOR constraints. The sequel follows Sections 3-4 in [36].

Definitions Formally, let $\mathbf{F}$ denote a Boolean formula, V the total number of variables, and F a satisfying variable assignment of $\mathbf{F}$. An individual XOR constraint over variables $\mathbf{X}$ has the form $\bigotimes_{i\in[1,\ V]} b_i \cdot X_i = b_0$, where $b_{0|i} \in \{0, 1\}$. The coefficients b_i determine the variables involved in the constraint, whereas the *parity bit* b_0 determines whether an even or an odd number of variables must be set to 1. For example, given $V = 4$, the XOR constraint $x_1 \otimes x_3 \otimes x_4 = 1$ is satisfied if one or three of the involved variables are set to 1, e.g., by the assignments $\mathbf{x_1} = 1 \wedge x_2 = 1$ or $\mathbf{x_1} = 1 \wedge \mathbf{x_3} = 1 \wedge \mathbf{x_4} = 1$, but not by the assignments $\mathbf{x_1} = 1 \wedge \mathbf{x_3} = 1 \wedge x_2 = 1$ or $\mathbf{x_3} = 1 \wedge \mathbf{x_4} = 1$ (omitted variables are equal to 0). Within WeightGen, XOR constraints serve a domain-independent, technical purpose (see below).

Furthermore, let $\omega(\cdot)$ denote a black-box weight function that for each F returns a number in $(0,\ 1]$; and ω_{min} (resp. ω_{max}) the minimal (resp. maximal) weight over all satisfying assignments of $\mathbf{F}$. The weight function induces the probability distribution over satisfying assignments of $\mathbf{F}$, where $P_\omega(F) = \omega(F)/\sum \omega(F')$. Quantity $r = \omega_{max}/\omega_{min}$ is the (possibly unknown) *tilt* of the distribution induced by $\omega(\cdot)$. Given a user-provided upper bound on tilt $\hat{r} \geq r$ and a desired sampling error tolerance $\kappa \in (0,\ 1)$ (the lower κ, the tighter the bounds on the sampling error), WeightGen generates a random solution F. Performance

guarantees concern both accuracy and efficiency of the algorithm and depend on the parameters and the total number of variables V; see Section 4.5 for details.

Algorithm Recall that the core idea that underlies sampling with guarantees is partitioning the overall solution space into a number of random cells by adding random XOR constraints. WeightGen proceeds in two phases: 1) the estimation phase and 2) the sampling phase. The goal of the *estimation phase* is to estimate the number of XOR constraints necessary to obtain a "small" cell, where the maximal cell weight is determined by the desired sampling tolerance.

The *sampling phase* starts with applying the estimated number of XOR constraints. If it obtains a cell whose total weight lies within a certain range, which depends on κ, a solution is sampled exactly from all solutions in the cell; otherwise, it adds a new random XOR constraint. However, the number of XOR constraints that can be added is limited. If the algorithm cannot obtain a suitable cell, it indicates failure and returns no sample.

Both phases make use of a *bounded* oracle that terminates as soon as the total weight of enumerated solutions exceeds a predefined number. It enumerates solutions of the original problem $\mathbf{F}$ augmented with the XOR constraints. Together, m XOR constraints identify one cell belonging to a partitioning of the overall solution space into 2^m cells.

The core operation of WeightGen involves drawing coefficients uniformly at random, which induces a random partitioning of the solution space that satisfies the 3-*wise independence property*, i.e., knowing the cells for two arbitrary assignments does not provide any information about the cell for a third assignment [66]. This ensures desired statistical properties of random partitions, required for the theoretical guarantees. Algorithm 3 shows the pseudocode for WeightGen. It is structured similarly to that of UniGen2, a close cousin of WeightGen [35]. Lines 1-3 correspond to the estimation phase and Lines 4-8 correspond to the sampling phase. SolveBounded stands for the bounded enumeration oracle.

The parameters of the estimation phase are fixed to particular theoretically motivated values (see Chakraborty et al. [36, Section 4]). $pivot_{est}$ denotes the maximal weight of a cell at the estimation phase; $pivot_{est} = 46$ corresponds to estimation error tolerance $\varepsilon_{est} = 0.8$ (Line 10). If the total weight of solutions in a given cell exceeds $pivot_{est}$, a new random XOR constraint is added in order to eliminate a number of solutions. Repeating the process for a number of iterations increases the confidence of the estimate, e.g., 17 iterations result in $1 - \delta_{est} = 0.8$ (Line 1). Note that Estimate essentially estimates the total weight of *all* solutions, from which N_{XOR}, the initial number of XOR constraints for the sampling phase, is derived (Line 4).

Algorithm 3 WeightGen [36]

Input: Boolean formula F, weight ω, tilt bound $\hat{r}$, sampling error tolerance parameter κ

Output: $\omega(\cdot) \in [1/\hat{r},\ 1]$, bounded enumeration algorithm SolveBounded

1: **for** 17 iterations **do** ▷ Corresponds to $\delta_{est} = 0.2$
2: $WeightEstimates \overset{Add}{\leftarrow}$ Estimate($\emptyset$)
3: $TotalWeight =$ Median($WeightEstimates$)
4: $N_{XOR} = \mathcal{O}\left(\log_2 TotalWeight/\left(1+\kappa^{-1}\right)\right)$
5: $loThresh \propto (1+\kappa)/\kappa^2$, $hiThresh \propto (1+\kappa)^3/\kappa^2$
6: **for** $N_{samples}$ times **do**
7: $InitXORs = \{$RandomXOR() $\times N_{XOR}$ times$\}$
8: Generate(κ, [$loThresh$, $hiThresh$], $InitXORs$, 3)

9: **function** Estimate($XORs$)
▷ Returns an estimate of the total weight of all solutions
10: $pivot_{est} = 46$ ▷ Corresponds to $\varepsilon_{est} = 0.8$
11: $Sols \leftarrow$ SolveBounded(F, $XORs$, $pivot_{est}$)
12: $CellWeight \leftarrow \sum_{s \in Sols} \omega(s)$
13: **if** $CellWeight \leq pivot_{est}$ **then** ▷ Cell of the "right" size
14: **return** $CellWeight \times 2^{|XORs|}$
15: **else** ▷ Shrink cell by adding XOR constraint
16: Estimate($XORs \cup$ RandomXOR())

17: **function** Generate(κ, [$loThresh$, $hiThresh$], $XORs$, i)
▷ Returns a random solution of F
18: $Sols \leftarrow$ SolveBounded(F, $XORs$, $hiThresh$)
19: $CellWeight \leftarrow \sum_{s \in Sols} \omega(s)$
20: **if** $CellWeight \in$ [$loThresh$, $hiThresh$] **then** ▷ Cell of the "right" size
21: **return** SampleExactly($Sols$, ω)
22: **else if** $CellWeight > loThresh \wedge i > 0$ **then** ▷ Cell is too large
23: Generate(κ, [$loThresh$, $hiThresh$],$XORs \cup$ RandomXOR(), $i-1$)
24: **else** ▷ Cell is too small
25: **return** $\perp$

A similar procedure is employed at the sampling phase. It starts with N_{XOR} constraints and adds at most *three* extra constraints. The user-chosen error tolerance parameter κ determines the range [$loThresh$, $hiThresh$], within which the total weight of a suitable cell should lie (Line 5). For example, $\kappa = 0.9$ corresponds to range [6.7, 49.4]. If a suitable cell can be obtained, a solution is sampled exactly from all solutions in the cell; otherwise, no sample is returned.

Requiring the total cell weight to exceed a particular value ensures the lower bound on the sampling accuracy.

The preceding presentation makes two simplifying assumptions: (1) all weights lie in $[1/r, 1]$; (2) adding XOR constraints never results in unsatisfiable subproblems (empty cells). The former is relaxed by multiplying pivots by $\hat{\omega}_{max} = \hat{\omega}_{min} \times \hat{r} < 1$, where $\hat{\omega}_{min}$ is the smallest weight observed so far. The latter is solved by simply restarting an iteration with a newly generated set of constraints. See Chakraborty et al. [36] for the full explanation, including the precise formulae to compute all parameters.

4.5 Flexics: Flexible sampler with guarantees

In this section, we present Flexics, a pattern sampler that uses WeightGen as the umbrella sampling procedure. To this end, we 1) extend it to CSPs with binary variables, a class of problems that is more general than SAT and that includes pattern mining as described in Section 4.4; 2) augment existing pattern mining algorithms for use with WeightGen; and 3) investigate the properties of pattern quality measures in the context of WeightGen's requirements.

WeightGen was originally presented as an algorithm to sample solutions of the SAT problem. Pattern mining problems cannot be efficiently tackled by pure Boolean solvers due to the prominence of cardinality constraints (e.g., $minfreq$). However, we observe that the core sampling procedure is applicable to any CSP with binary variables, as its solution space can be partitioned with XOR constraints in the required manner.

Based on this insight, we present two variants of Flexics that differ in their oracles. Each oracle is essentially a pattern mining algorithm extended to support XOR constraints along with common constraints on patterns. The first one, dubbed GFlexics, builds upon the generic formalization and solving techniques described in Section 4.4 and thus supports a wide range of constraints. Owing to the properties of the *coverage* constraint, XOR constraints only need to involve item variables[1], which makes them relatively short, mitigating the computational overhead. Moreover, this perspective helps us design the second approach, dubbed EFlexics, which uses an extension of Eclat [153], a well-known mining algorithm, as an oracle. It is tailored for a single task (frequent itemset mining, i.e., it only supports the $minfreq$ constraint), but is capable of handling larger datasets. We describe each oracle in detail in the following subsections.

[1] In other words, item variables I are the *independent support* of a pattern mining CSP.

Given a dataset $\mathcal{D}$, constraints $\mathcal{C}$, a quality measure φ, and the error tolerance parameter $\kappa \in (0,\ 1)$, FLEXICS first constructs a CSP corresponding to the task of mining patterns satisfying $\mathcal{C}$ from $\mathcal{D}$. It then determines parameters for the sampling procedure, including the appropriate number of XOR constraints, and starts generating samples. To this end, it uses one of the two proposed oracles to enumerate patterns that satisfy $\mathcal{C}$ *and* random XOR constraints. Both variants of FLEXICS support sampling from black-box distributions derived from quality measures and, most importantly, preserve the theoretical guarantees of WEIGHTGEN[2]:

Theorem 1. *The probability that* FLEXICS *samples a random pattern* p *that satisfies constraints* $\mathcal{C}$ *from a dataset* $\mathcal{D}$*, lies within a bounded range determined by the quality of the pattern* $\varphi(p)$ *and* κ*:*

$$\frac{\varphi(p)}{Z_\varphi} \times \frac{1}{1+\varepsilon(\kappa)} \leq P(\textsc{Flexics}(\mathcal{D},\mathcal{C},\varphi;\kappa)=p) \leq \frac{\varphi(p)}{Z_\varphi} \times (1+\varepsilon(\kappa))$$

Proof. Theorem 3 of [36] states:

$$P_\omega(F)/(1+\varepsilon(\kappa)) \leq \hat{P}_F \leq P_\omega(F) \times (1+\varepsilon(\kappa))$$

where $\hat{P}_F$ denotes the probability that WEIGHTGEN called with parameters $\hat{r}$ and κ samples the solution F, $P_\omega(F) \propto \omega(F)$ denotes the target probability of F, and $\varepsilon(\kappa) = (1+\kappa)\left(2.36 + 0.51/(1-\kappa)^2\right) - 1$ denotes sampling error derived from κ.

For technical purposes, we introduce the notion of the *weight* of a pattern as its quality scaled to the range $(0,\ 1]$, i.e., $\omega_\varphi(p) = \varphi(p)/C$, where C is an arbitrary constant such that $C \geq \max_{p\in\mathcal{L}} \varphi(p)$. The proof follows from Theorem 3 of [36] and the observation that $\textsc{Flexics}(\mathcal{D},\mathcal{C},\varphi;\kappa)$ is equivalent to $\textsc{WeightGen}(\text{CSP}(\mathcal{D},\mathcal{C}),\omega_\varphi;\kappa)$. The estimation phase effectively corrects for potential discrepancy between C and Z_φ. □

Furthermore, Theorem 4 of Chakraborty et al. [36] provides *efficiency guarantees*: the number of calls to the oracle is linear in $\hat{r}$ and polynomial in $|\mathcal{I}|$ and $1/\varepsilon(\kappa)$. The assumption that the tilt is *bounded from above* by a reasonably low number is the only assumption regarding a (black-box) weight function. Moreover, it only affects the efficiency of the algorithm, but not its accuracy.

[2]Theorem 1 corresponds to and follows from Theorem 3 of [36].

Thus, using a quality measure with FLEXICS requires knowledge of two properties: scaling constant C and tilt bound $\hat{r}$. In practice, both are fairly easy to come up with for a variety of measures. For example, for $freq$ and $purity$, $C = |\mathcal{D}|$, $\hat{r} = \theta^{-1}$ and $C = 1$, $\hat{r} = 2$ respectively; see Section 4.7 for another example.

Example Assume that FLEXICS is requested to sample patterns from the dataset in Table 4.3 with the following parameters: $\mathcal{C} = minfreq\,(0.4)$, which results in 63 frequent patterns; $\varphi = uniform$ (i.e., $\varphi\,(p) \equiv 1$), and $\kappa = 0.9$. The estimation phase returns $TotalWeight = 61$, an accurate estimate of the number of frequent patterns; then $N_{XOR}\,(TotalWeight, \kappa) = 0$. In practice, obtaining a suitable cell (a random subset of all frequent patterns), whose total weight lies between $loThresh\,(\kappa) = 6.7$ and $hiThresh\,(\kappa) = 49.4$, requires posting one additional random XOR constraint.

1	2	3	4		6		
		3	4	5		7	
1	2	3	4			7	
1	2	3			6		8
1	2	3	4		6	7	8

Table 4.3: A toy dataset.

For example, the XOR constraint $I_6 \otimes I_7 = 1$ yields a cell with 40 patterns (e.g., $\{1, 2, 6\}$ or $\{3, 4, 7\}$), each of which can be sampled from that cell with probability $1/40$ as the eventual sample returned by FLEXICS. The constraint $I_1 \otimes I_2 \otimes I_6 \otimes I_8 = 0$ yields a cell with 31 patterns (e.g., $\{2, 8\}$ or $\{3, 4, 7\}$ again). Enumerating all patterns that satisfy a combination of pattern constraints $\mathcal{C}$ and random XOR constraints is the key technical challenge within FLEXICS, which we address in the following sections.

GFlexics: Generic pattern sampler

The first variant relies on CP4IM [68], a constraint programming-based mining system. A wide range of constraints supported by CP4IM are automatically supported by the sampler and can be freely combined with various quality measures.

In order to turn CP4IM into a suitable bounded oracle, we need to extend it with an efficient propagator for XOR constraints. This propagator is based on the process of *Gaussian elimination* [65], a classical algorithm for solving systems of linear equations. Each XOR constraint can be viewed as a linear equality over the field $\mathbb{F}_2$ of two elements, 0 and 1, and all coefficients form a binary matrix (Figure 4.1.2). At each step, the matrix is updated with the latest variable assignments and transformed to *row echelon form*, where all ones are on or above the main diagonal and all non-zero rows are above any

$x_1 \otimes x_5 = 1$	1 0 0 0 1 | 1	1 0 0 0 1 | 1	1 0 0 0 1 | 1	→ **0** 0 0 0 **0** | **1**
$x_2 \otimes x_3 \otimes x_4 \otimes x_5 = 0$	0 1 1 1 1 | 0	→ 0 **1** 0 0 0 | **0**	0 0 0 1 1 | 1	**0** 0 0 1 **0** | **0**
$x_1 \otimes x_2 \otimes x_3 \otimes x_5 = 0$	1 1 1 0 1 | 0	→ 0 0 **1** 0 0 | **1**	0 0 0 0 0 | 0	0 0 0 0 0 | 0
$x_2 \otimes x_4 \otimes x_5 = 1$	0 1 0 1 1 | 1	0 0 0 1 1 | 1	0 0 0 0 0 | 0	0 0 0 0 0 | 0
1) Random XOR constraints	2) Initial constraint matrix	3) Echelonized matrix: assignments $x_2 = 0$ and $x_3 = 1$ are derived	4) Updated matrix (rows 2 and 4 are swapped)	5) If x_1 and x_5 are set to 1 (e.g., by search), the system is unsatisfiable

Figure 4.1: Propagating XOR constraints using Gaussian elimination in $\mathbb{F}_2$.

rows of all zeroes (Figure 4.1.3). During echelonization, two situations enable propagation. If a row becomes empty while its right hand side is equal to 1, the system is unsatisfiable and the current search branch terminates (Figure 4.1.5). If a row contains only one free variable, it is assigned the right hand side of the row (Figure 4.1.3).

Gaussian elimination in $\mathbb{F}_2$ can be performed very efficiently, because no division is necessary (all coefficients are 1), and subtraction and addition are equivalent operations. For a system of k XOR constraints over n variables, the total time complexity of Gaussian elimination is $\mathcal{O}\left(k^2 n\right)$.

EFlexics: Efficient pattern sampler

Generic constraint solvers currently cannot compete with the efficiency and scalability of specialized mining algorithms. In order to develop a less flexible, yet more efficient version of our sampler, we extend the well-known Eclat algorithm to handle XOR constraints. Thus, EFlexics is tailored for frequent itemset sampling and uses EclatXOR (Algorithm 4) as an oracle.

Algorithm 4 shows the pseudocode of the extended Eclat. The algorithm relies on the *vertical* data representation, i.e., for each candidate item, it stores a set of indices of transactions (TIDs), in which this item occurs (Line 4). Eclat starts with determining frequent items and ordering them, typically by frequency ascending. It explores the search space in a depth-first manner, where each branch corresponds to (ordered) itemsets that share a prefix.

The core operation is referred to as *processing an equivalence class* of itemsets (EqClass). For each prefix, Eclat maintains a set of candidate suffixes, i.e.,

Algorithm 4 ECLAT augmented with XOR constraint handling (Lines 16-22)

Input: Dataset $\mathcal{D}$ over items $\mathcal{I}$, min.freq θ, XOR matrix M
Output: Item order $\succ_{\mathcal{I}}$ by frequency ascending

1: **function** ECLATXOR($\mathcal{D}$, θ, M)
▷ *Mine all frequent patterns that satisfy XOR constraints encoded by M*
2: Frequent items $FI = \emptyset$
3: **for** item $i \in \mathcal{I}$ **do**
4: $TID_i = \{\text{transaction index } t \in \mathcal{T} \mid \mathcal{D}_{ti} = 1\}$
5: **if** $|TID_i| \geq \theta$ **then** ▷ Item is frequent
6: $FI \overset{Add}{\leftarrow} (i, TID_i)$
7: SORT(FI, $\succ_{\mathcal{I}}$)
8: **for** $i \in FI$ **do**
9: Candidate suffixes $CS = \{i' \in FI \setminus i \mid i' >_{\mathcal{I}} i\}$
10: EQCLASS($\{i\}$, CS, M)
11: **function** EQCLASS(Prefix P, cand.suffixes $CS \neq \emptyset$, M)
▷ *Mine all patterns that start with P*
12: **if** CHECKCONSTRAINTS(P, M) **then**
13: **return** P ▷ Return prefix, if it satisfies XORs
14: **for** candidate suffix $s \in CS$ **do**
15: $P' = P \cup s$; frequent suffixes $FS = \{f \in CS \setminus s \mid f >_{\mathcal{I}} s \wedge |f.TID \cap s.TID| \geq \theta\}$
▷ *Propagate XOR constraints*
16: $U_1 = \{s\}$, $U_0 = CS \setminus FS$ ▷ Variable updates
17: $M' =$ UPDATEANDECHELONIZE(M, U_1, U_0)
18: $(A_1, A_0) =$ PROPAGATE(M') ▷ Item variables
▷ that were assigned value 1 or 0 by propagation
19: $FS' = FS \setminus (A_1 \cup A_0)$
20: **if** $A_1 \neq \emptyset$ **then** ▷ If prefix was extended,
▷ update TIDs and check support
21: $P' \leftarrow P' \cup A_1$, $\Delta_{TID} = \bigcap_{f \in A_1} f.TID$
22: $FS' \leftarrow \{f' \in FS' : |f'.TID \cap \Delta_{TID}| \geq \theta\}$
23: **if** $|P'.TID| \geq \theta \wedge FS' \neq \emptyset$ **then**
24: EQCLASS(P', FS'', M')

items that follow the last item of the prefix in the item order and are frequent. The frequency of a candidate suffix, given the prefix, is computed by intersecting its TID with the TID of the prefix (Lines 9, 15, and 22).

We extend ECLAT with XOR constraint handling (Lines 16-22). Variable updates

stem from ECLAT extending the prefix and removing infrequent suffixes (Line 16). XOR propagation can result in extending the prefix or removing candidate suffixes as well (Line 19). Furthermore, if the prefix has been extended, TIDs of candidate suffixes need to be updated, with some of them possibly becoming infrequent, leading to further propagation (Lines 19-22). If the prefix becomes infrequent, the search branch terminates.

Fixed variable-order search, like ECLAT, is an advantageous case for Gaussian elimination [130]: non-zero elements are restricted to the right region of the matrix, hence Gaussian elimination only needs to consider a contiguous, progressively shrinking subset of columns. Total memory overhead of ECLATXOR compared to plain ECLAT is $\mathcal{O}\left(d \times |\mathcal{F}| \times N_{XOR} + pivot \times r\right)$, where d denotes maximal search depth, $|\mathcal{F}|$ the number of frequent singletons (columns of a matrix), and N_{XOR} the number of XOR constraints (rows of a matrix). The first term refers to a set of XOR matrices in unexplored search branches, whereas the second term refers to storing itemsets in a cell (Line 19 in Algorithm 3).

WeightGen: implementation details

Following suggestions of [35], we implement *leapfrogging*, a technique that improves the performance of the umbrella sampling procedure and thus benefits both GFLEXICS and EFLEXICS. First, after three iterations of the estimation phase, we initialize the following iterations with a number of XOR constraints that is equal to the smallest number returned in the previous iterations (rather than with zero XORs). Second, in the sampling phase, we start with one XOR constraint more than the number suggested by theory. If the cell is too small, we remove one constraint; if it is too large, we proceed adding (at most two) constraints. Both modifications are based on the observation that theoretical parameter values address hypothetical corner cases that rarely occur in practice. Finally, we only run the estimation phase until the initial number of XOR constraints, which only depends on the median of total weight estimates, converges. For example, if the estimation phase is supposed to run for 17 iterations, the convergence can happen as early as after 9 iterations.

4.6 Pattern sampling with Flexics – an example

In this section we illustrate the workings of the FLEXICS sampler, in particular focusing on the effects of XOR constraints. In order to visualize large pattern collections, we obtain a two-dimensional feature representation for patterns

using *principal component analysis* in the following manner: 1) we mine all patterns that satisfy constraints $\mathcal{C}$; 2) we construct the *pattern description matrix*, where each column corresponds to an item, each row corresponds to a pattern, and an entry is equal to 1, if the given pattern contains the given item, and 0 otherwise; 3) we use the two principal components obtained from this matrix to visualize the patterns. Thus, two patterns are close to each other in the transformed space if their descriptions are similar, i.e., if they share many items. The transformation does not consider the quality of the patterns.

We use the following setting: the `vote` dataset (see Section 4.8 for details), constraints $\mathcal{C} = minfreq\,(0.09) \wedge closed \wedge minlen\,(7)$, which result in 19 530 patterns, and $\varphi = freq$. Figure 4.2 shows that the pattern space comprises several "clusters." The key challenge in sampling is balancing the relative importance of these "clusters" *and* patterns within each "cluster." In the sequel, we describe how FLEXICS tackles these challenges. We use the size of the dataset as the scaling constant, i.e., $C = |\mathcal{D}| = 435$ and thus $\omega\,(p) = freq\,(p)/435$.

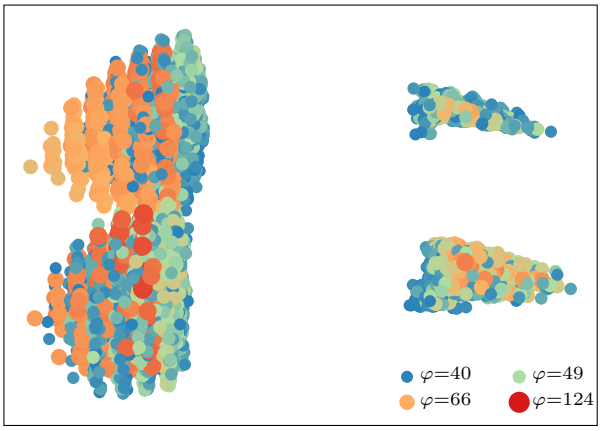


Figure 4.2: All patterns in the `vote` dataset that satisfy the constraints $\mathcal{C} = minfreq\,(0.09) \wedge closed \wedge minlen\,(7)$. The coordinates correspond to the two principal components of the pattern description matrix. The size and the color of a point indicate the value of $\varphi = freq$ of the corresponding pattern.

	Patterns satisfying $\mathcal{C} \wedge XORs$			
	Enumerated		Total	
$\lvert XORs \rvert$	Number	$\sum \omega(\cdot)$	Number	$\sum \omega(\cdot)$
1	458	46.04	9 850	1 263.43
2	435	46.04	4 956	634.93
3	385	46.04	2 480	317.94
4	405	46.05	1 264	161.23
5	365	46.09	627	79.23
6	310	39.05	310	39.05

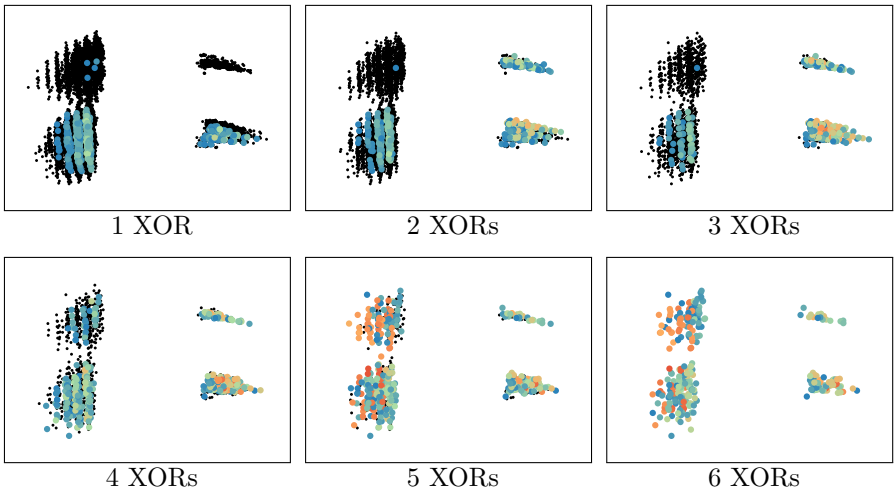


Figure 4.3: Obtaining a suitable cell (i.e., for the estimation phase, a cell with a total weight below $pivot_{est} = 46$) requires adding 6 XORs to the original constraints. Each additional XOR eliminates roughly a half of all solutions (black dots), which are spread uniformly across the "clusters." The number of solutions that need to be enumerated by the bounded oracle (colored dots) does not vary considerably.

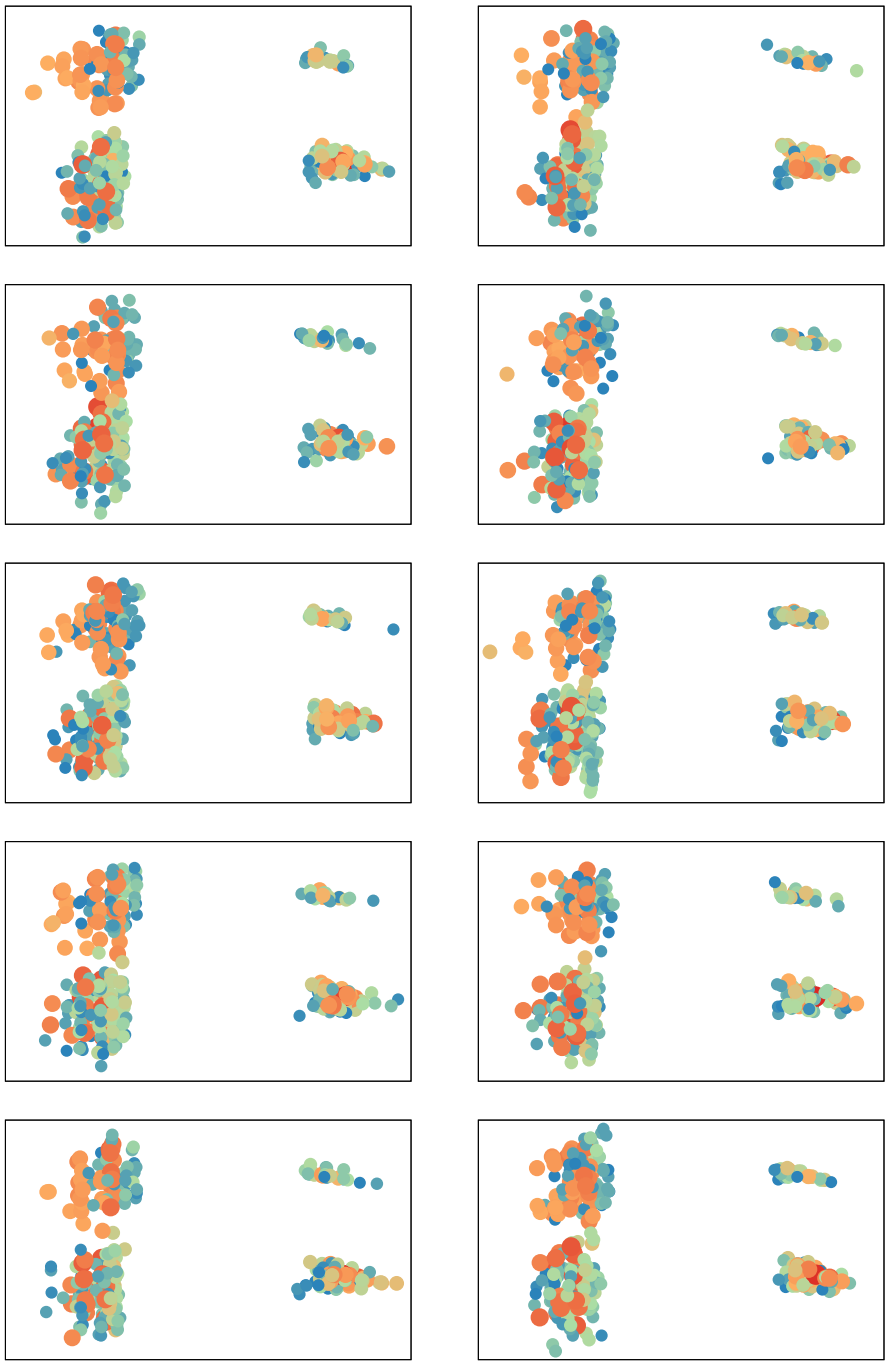


Figure 4.4: Ten random cells obtained with FLEXICS. The cells are different from each other. Each cell covers every "cluster" relative to its importance and contains diverse patterns within each "cluster." This ensures the diversity among sampled patterns.

Figure 4.3 illustrates how FLEXICS obtains a suitable cell by incrementing the number of XOR constraints and calling the bounded oracle. For example, at the estimation phase, the maximal size of a suitable cell, denoted $pivot_{est}$ (see Section 4.4), is equal to 46. FLEXICS starts by enumerating the solutions of the original problem without any XOR constraints. When the total weight of enumerated solutions exceeds 46, it adds the first random XOR, which eliminates roughly a half of all solutions (9 680 out of 19 530). The bounded oracle enumerates 458 solutions of the problem augmented with one XOR until the total weight exceeds 46 again, which implies that another XOR needs to be added. The solutions that are eliminated by additional XOR are spread uniformly across the "clusters", which illustrates the 3-wise independence property of random XOR constraints. A suitable cell is obtained with 6 XORs, which yields the following estimate of the total weight of all solutions: $39.05 \times 2^6 = 2499.2$, where 39.05 is the weight of a random cell and 2^6 is the number of cells in a partitioning induced by 6 XORs. The actual value is 2504.2, thus, the error of the estimation phase is well below the theoretical factor of 1.8. Figure 4.4 shows ten random cells obtained at the sampling phase. The cells are different from each other; moreover, each contains diverse patterns from every "cluster." This ensures the independence and the diversity of samples.

4.7 Pattern set sampling

We highlight the flexibility of FLEXICS by introducing and tackling the novel task of *sampling sets of patterns*. For the purposes of sampling, a set of patterns is essentially treated as a composite pattern. Typically, constituent patterns are required to be different from each other. The quality (and hence, the sampling probability) of a pattern set depends on collective properties of constituent patterns. These characteristics, coupled with the immense size of the pattern set search space, make sampling even more challenging.

To develop a sampler, we extend GFLEXICS with the CSP-formulation of the k-pattern set mining task [69], which in turn builds upon the formulation of the itemset mining task described in Section 4.4. Recall that a CSP is defined by a set of variables and constraints over these variables. Each constituent pattern is modeled with distinct item and transaction variables, i.e., I_{ik} and T_{tk} for the kth pattern p_k. Note that this increases the length of XOR constraints, which poses an additional challenge from the sampling perspective.

Any single-pattern constraint can be enforced for a constituent pattern, e.g., $minfreq\,(\theta)$, *closed*, or $minlen\,(\lambda)$. A common pattern set-specific constraint

is *no overlap*, which enforces that neither the itemsets (1), nor the sets of transactions that they cover (2) overlap:

$$(1)\ \forall i \in \mathcal{I}\ \sum I_{ik} \leq 1 \qquad (2)\ \forall t \in \mathcal{T}\ \sum T_{tk} \leq 1$$

Furthermore, there is typically a symmetry-breaking constraint that requires that the set of transaction indices of p_i lexicographically precedes those of $\{\, p_j \mid j > i \,\}$. This approach allows modeling a wide range of pattern set sampling tasks, e.g., sampling k-term DNFs, conceptual clusterings, redescriptions, and others. In this chapter, we use the problem of *tiling datasets* [57] as an example.

The main aim of tiling is to cover a large number of 1s in a binary 0/1 dataset with a given number of patterns. Thus, *a tiling* is essentially a set of itemsets that together describe as many item occurrences as possible. Without loss of generality, we describe the task of sampling non-overlapping 2-tilings ($k = 2$). Let p_1 and p_2 denote the constituent patterns of a 2-tiling. The quality of a tiling is equal to its *area*, i.e., the number of 1s that it covers:

$$area\left(\{p_1, p_2\}\right) = \left(freq\left(p_1\right) \times |p_1| + freq\left(p_2\right) \times |p_2|\right)$$

The scaling constant for *area* is $C = \sum \mathcal{D}_{ti}$, i.e., the total number of 1s in the dataset. The tilt bound is estimated as $\hat{r} = \sum \mathcal{D}_{ti} / \left(2 \times \left(|\mathcal{D}| \times \theta\right) \times \lambda\right)$, where the denominator is the smallest possible area of a 2-tiling given the constraints.

4.8 Experiments

The experimental evaluation focuses on accuracy, scalability, and flexibility of the proposed sampler. The research questions are as follows:

Q1 *How close is the empirical sampling distribution to the target distribution?*

Q2 *How does* Flexics *compare to the specialized alternatives?*

Q3 *Does* Flexics *scale to large datasets?*

Q4 *How flexible is* Flexics*, i.e., can it be used for new pattern sampling tasks?*

The implementations of GFlexics and EFlexics[3] are based on cp4im[4] and a custom implementation of Eclat respectively. Both are augmented with a

[3] Available at `https://bitbucket.org/wxd/flexics`.

[4] `https://dtai.cs.kuleuven.be/CP4IM`

	Constraints $\mathcal{C}$	Itemsets per dataset	Quality measure φ	Tilt bound $\hat{r}$
F	$min\mathbf{F}req\,(\theta)$	$\sim 60\,000$	$uniform\ (\varphi \equiv 1)$	1
FCL	$min\mathbf{F}req\,(\theta) \wedge$ $\mathbf{C}losed \wedge min\mathbf{L}en\,(\lambda)$	$\geq 15\,000$	$purity$	2
			$freq$	θ^{-1}

Table 4.4: Combinations of two constraint sets and three quality measures yield six experimental settings per dataset for sampling accuracy experiments; see Section 4.4 for definitions.

propagator for a system of XOR constraints based on the implementation of Gaussian elimination in the M4RI library[5] [96]. All experiments were run on a Linux machine with an Intel Xeon CPU@3.2GHz and 32Gb of RAM.

Q1: Sampling accuracy We study the sampling accuracy of GFLEXICS in settings with tight constraints, which yield a relatively low number of solutions. This allows us to compute the exact statistical distance between the empirical sampling distribution and the target distribution. We investigate settings with various quality measures and constraint sets as well as the effect of the tolerance parameter κ.

We select several datasets from the CP4IM repository[6] in the following way. For each dataset, we construct two constraint sets (see Table 4.4). We choose a value of θ such that there are approximately 60 000 frequent patterns. Given θ, we choose a value of $\lambda \geq 2$ such that there are at least 15 000 closed patterns that satisfy the *minlen* constraint. In order to obtain sufficiently challenging sampling tasks, we omit the datasets where the latter condition does not hold (i.e., there are too few closed "long" patterns). Combining two constraint sets with three quality measures yields six experimental settings per dataset. Table 4.5 shows dataset statistics and parameter values. For each $\kappa \in \{0.1, 0.5, 0.9\}$, we request 900 000 samples.

Let T denote the set of all itemsets that satisfy the constraints, E denote the multiset of all samples, and $\mathbf{1}_S$ its multiplicity function. For a given quality measure φ, target and empirical probabilities of sampling an itemset p are respectively defined as $P_T(p) = \varphi(p) / \sum_{p' \in T} \varphi(p')$ and $P_E(p) = \mathbf{1}_E(p)/|E|$. We use *Jensen-Shannon (JS) divergence* to quantify the statistical distance between P_T and P_E. Let $D_{KL}(P_1 \| P_2)$ denote the well-known *Kullback-Leibler*

[5] `https://bitbucket.org/malb/m4ri/`
[6] Source: `https://dtai.cs.kuleuven.be/CP4IM/datasets/`

	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	Density	θ	λ	$\lvert\mathcal{P}_\mathcal{C}\rvert$ F	$\lvert\mathcal{P}_\mathcal{C}\rvert$ FCL
`german`	1000	112	34%	0.35 (349)	2	61 074	16 576
`heart`	296	95	47%	0.43 (127)	2	59 304	15 487
`hepatitis`	137	68	50%	0.39 (53)	5	65 662	19 450
`kr-vs-kp`	3196	74	49%	0.69 (2190)	6	62 462	22 471
`primary`	336	31	48%	0.09 (30)	7	63 209	19 296
`splice`	3190	287	21%	0.04 (122)	3	60 957	33 654
`vote`	435	48	33%	0.09 (40)	7	63 340	19 530

Table 4.5: Datasets used in sampling experiments. $\lvert\mathcal{P}_\mathcal{C}\rvert$ denotes the number of patterns that satisfy constraints $\mathcal{C}$.

divergence between distributions P_1 and P_2. JS-divergence D_{JS} is defined as follows:

$$D_{JS}\left(P_T \| P_E\right) = 0.5 \times \left(D_{KL}\left(P_T \| P_M\right) + D_{KL}\left(P_E \| P_M\right)\right)$$

$$\text{where } P_M = 0.5 \times \left(P_T + P_E\right)$$

JS-divergence ranges from 0 to 1 and, unlike KL-divergence, does not require that $P_T(p) > 0 \Rightarrow P_E(p) > 0$, i.e., that each solution is sampled at least once, which does not always hold in sampling experiments. We compare D_{JS} attained with our sampler with that of the ideal sampler, which materializes all itemsets satisfying the constraints, computes their qualities, and uses these to sample directly from the target distribution.

A characteristic experiment in detail Our experiments show that results are consistent across various datasets. Therefore, we first study the results on the `vote` dataset in detail. Table 4.6 shows that the theoretical error tolerance parameter κ has no considerable effect on practical performance of the algorithm, except for runtime, which we evaluate in subsequent experiments. One possible explanation is the high quality of the output of the estimation phase, which thus alleviates theoretical risks that have to be accounted for in the general case (see below for a numerical characterization). Hence, in the following experiments we use $\kappa = 0.9$ unless noted otherwise.

JS-divergences for different quality measures and constraint sets are impressively low, equivalent to the highest possible sampling accuracy attainable with the ideal sampler. Figure 4.5 illustrates this for $minfreq\,(0.09) \wedge closed \wedge minlen\,(7)$, $\varphi = freq$, and $\kappa = 0.9$ ($D_{JS} = 0.004$): the sampling frequency of an average

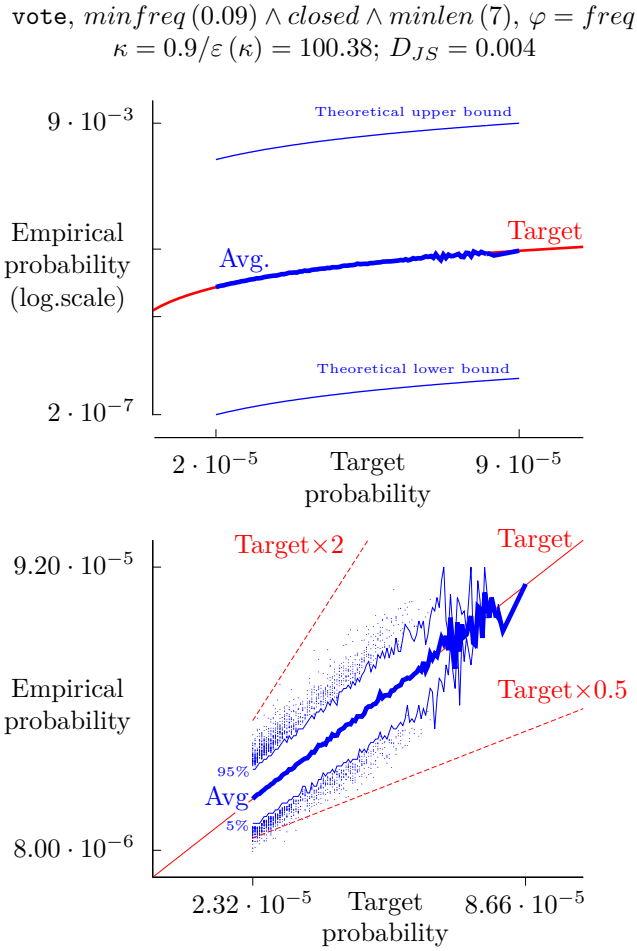


Figure 4.5: **Top**: The theoretical guarantees for the high value of the error tolerance parameter $\kappa = 0.9$ allow the empirical frequency for a given pattern to deviate from the target probability by a factor $1 + \epsilon\,(\kappa) = 101.38$, whereas in practice, the sampling error is considerably lower. On average, frequencies are close to the target probabilities. **Bottom**: Empirical sampling frequencies of itemsets that share the same target probability, i.e., have the same quality. 90% of frequencies are well within a factor 2 from the target. (The dots show the tails of the empirical probability distribution for a given target probability.)

	vote dataset, JS-divergence from target					
	Uniform ($\hat{r}=1$)		Purity ($\hat{r}=2$)		Frequency ($\hat{r}=11$)	
κ	F	FCL	F	FCL	F	FCL
0.9	0.013	0.004	0.013	0.004	0.013	0.004
0.5	0.013	0.004	0.013	0.004	0.013	0.004
0.1	0.013	0.004	0.013	0.004	0.013	0.004
Ideal sampler	0.013	0.004	0.013	0.004	0.013	0.004

Table 4.6: Sampling accuracy of FLEXICS (here GFLEXICS) is consistently high across quality measures, constraint sets ($min\mathbf{F}req\,(0.09)$ vs. $min\mathbf{F}req\,(0.09) \wedge \mathbf{C}losed \wedge min\mathbf{L}en\,(7)$), and error tolerance κ. JS-divergence is impressively low, equivalent to that of the ideal sampler.

itemset is close to the target probability. For at least 90% of patterns, the sampling error does not exceed a factor of 2, which is two orders of magnitude lower than the factor $1 + \epsilon\,(\kappa) = 101.38$ allowed by the theoretical guarantees.

Table 4.7 shows that similar conclusions hold for several other datasets. Over all experimental settings, the error of the estimation of the total weight of all solutions, which is used to derive the number of XOR constraints for the sampling phase, never exceeds 10%, whereas the bounds assume the error of 45 to 80%. This helps explain why practical errors are considerably lower than theoretical bounds.

In line with theoretical expectations (see Section 4.5), the `splice` dataset proves the most challenging due to the large number of items (variables in XOR constraints). As a result, GFLEXICS does not generate the requested number of samples within the 24-hour timeout. We study the runtime in the following experiment.

Q2: Comparison with alternative pattern samplers We compare FLEXICS to ACFI [15] and TS [19], alternative samplers[7] described in Section 4.3, in the settings that they are tailored for. ACFI only supports the setting with a single $minfreq\,(\theta)$ constraint and $\varphi = uniform$. It is run with a burn-in of 100 000 steps and uses a built-in heuristic to determine the number of steps between consecutive samples. TS is evaluated in the setting with $\varphi = freq$ and both constraint sets from the previous experiments. It samples from two

[7]The code was provided by their respective authors. We also obtained the "unmaintained" code for the *uniform* LRW sampler (personal communication), but were unable to make it run on our machines. The code for the FCA sampler was not available (personal communication).

	JS-divergence, $\kappa = 0.9$					
	Uniform		Purity		Frequency	
	F	FCL	F	FCL	F	FCL
`german`	0.012	0.003	0.013	0.003	0.013	0.003
`heart`	0.012	0.003	0.012	0.003	0.012	0.003
`hepatitis`	0.013	0.004	0.014	0.004	0.013	0.004
`kr-vs-kp`	0.013	0.005	0.013	0.005	0.013	0.005
`primary`	0.013	0.004	0.013	0.004	0.013	0.004
`splice`	–	–	–	–	–	–
`vote`	0.013	0.004	0.013	0.004	0.013	0.004

Table 4.7: Results of sampling accuracy experiments. Even with high error tolerance $\kappa = 0.9$, JS-divergence of FLEXICS (here GFLEXICS) is consistently low across datasets, quality measures, and constraint sets. (On the `splice` dataset, GFLEXICS generates less than 900 000 samples before the timeout; see also Table 4.9.)

of the distributions it supports, $freq$ and $freq^4$; samples that do not satisfy the constraints are rejected. Both samplers are requested to generate 900 000 samples and are allowed to run up to 24 hours. Datasets and parameters are identical to the previous experiments.

Table 4.8 shows the accuracy of the samplers. The performance of FLEXICS is on par with specialized samplers. That is, in uniform frequent itemset sampling, the accuracy of both FLEXICS and ACFI is equivalent to that of the ideal sampler and can therefore not be improved. When sampling proportional to frequency, it is equivalent to the accuracy of the exact two-step sampler TS $\sim freq$. However, the latter does not directly take constraints into account, which poses considerable problems on most datasets. For example, for the `heart` dataset, TS fails to generate a single accepted sample, despite generating 2 billion unconstrained candidates. This issue is not solved by increasing the bias towards more frequent itemsets by sampling proportional to $freq^4$. Furthermore, this would substantially decrease accuracy, as seen in `primary` and `vote`.

Table 4.9 shows the runtimes for frequent itemset sampling (i.e., only the $minfreq$ constraint). In most settings, EFLEXICS provides runtime benefits over GFLEXICS. The `splice` dataset is the most challenging due to the large number of items; it highlights the importance of an efficient constraint oracle. Accordingly, the specialized sampler ACFI is from 6 to 22 milliseconds faster than a faster variant of FLEXICS in uniform sampling (excluding `splice`). In frequency-weighted sampling, FLEXICS is considerably faster in the settings with

	JS-divergence (for TS, acceptance rate)							
	Uniform		Frequency					
	F		F			FCL		
	GF	ACFI	GF	TS$\sim freq$	TS$\sim freq^4$	GF	TS$\sim freq$	TS$\sim freq^4$
`german`	0.01	0.01	0.01	–	–	0.00	–	–
				($9{\cdot}10^{-8}$)	(0.02)		($5{\cdot}10^{-8}$)	(0.06)
`heart`	0.01	0.01	0.01	–	–	0.00	–	–
				(0)	($4{\cdot}10^{-10}$)		(0)	($3{\cdot}10^{-3}$)
`hepatitis`	0.01	0.01	0.01	–	–	0.00	–	–
				($2{\cdot}10^{-6}$)	(0.01)		($1{\cdot}10^{-6}$)	(0.01)
`kr-vs-kp`	0.01	0.01	0.01	–	–	0.01	–	–
				($7{\cdot}10^{-7}$)	(0.01)		($4{\cdot}10^{-7}$)	($4{\cdot}10^{-3}$)
`primary`	0.01	0.01	0.01	0.01	0.40	0.01	0.01	0.27
				(0.30)	(0.99)		(0.10)	(0.13)
`splice`	–	0.01	–	–	–	–	–	–
				(0)	(0)		(0)	(0)
`vote`	0.01	0.01	0.01	0.01	0.23	0.00	0.01	0.14
				(0.13)	(0.94)		(0.05)	(0.22)

Table 4.8: The accuracy of FLEXICS (here GFLEXICS) is consistent across settings. In uniform frequent itemset sampling, performance of FLEXICS as well as of ACFI is equivalent to that of the ideal sampler (not shown). In frequency-weighted sampling, it is comparable to the exact two-step sampler (TS $\sim freq$) with rejection. However, the latter suffers from low acceptance rates, which, for settings marked with '–', is not improved by increasing bias (TS $\sim freq^4$). On `splice`, neither TS nor FLEXICS generate 900 000 samples before the timeout; see also Table 4.9.

tighter constraints, where the two-step sampler is slow to generate accepted samples. This illustrates the overhead as well as the benefits of the flexibility of the proposed approach. Furthermore, in these settings, there are at most 66 000 patterns, which is too low to suggest the need for pattern sampling (recall that the primary goal of these experiments was to evaluate and compare sampling accuracy) and does not allow for the overhead amortization. We therefore tackle settings with a substantially larger number of patterns in the following experiments.

Q3: Scalability To study scalability of the proposed sampler, we compare its runtime costs with those required to construct an ideal sampler with

	$\varphi = uniform$, $\mathcal{C} = \mathrm{F}$			$\varphi = freq$, $\mathcal{C} = \mathrm{F}$		
	GFlexics	EFlexics	ACFI	GFlexics	EFlexics	TS$\sim freq$
`german`	110	25	39	133	34	58540
`heart`	60	45	24	73	44	–
`hepatitis`	23	33	11	30	45	2632
`kr-vs-kp`	59	9	6	59	10	8731
`primary`	10	10	4	27	25	0.10
`splice`	170360	1376	580	–	1095	–
`vote`	25	19	8	46	28	0.03

Table 4.9: Runtime in milliseconds required to sample a frequent itemset, including pre-processing, i.e., estimation or burn-in, amortized over 1000 samples. Both variants of Flexics are suitable for anytime exploration, although slower than the specialized samplers. The two-step sampler is the fastest in the task it is tailored for, but fails in the settings with tighter constraints. EFlexics provides runtime benefits compared to GFlexics.

lcm[8], an efficient frequent itemset miner [136]. To this end, we estimate the costs of completing the following scenario: pre-processing (estimation or counting), followed by sampling 100 itemsets in two batches of 50. We use non-synthetic datasets from the FIMI repository[9], which have fewer than one billion transactions and select θ such that there are more than one billion frequent itemsets (see Table 4.10).

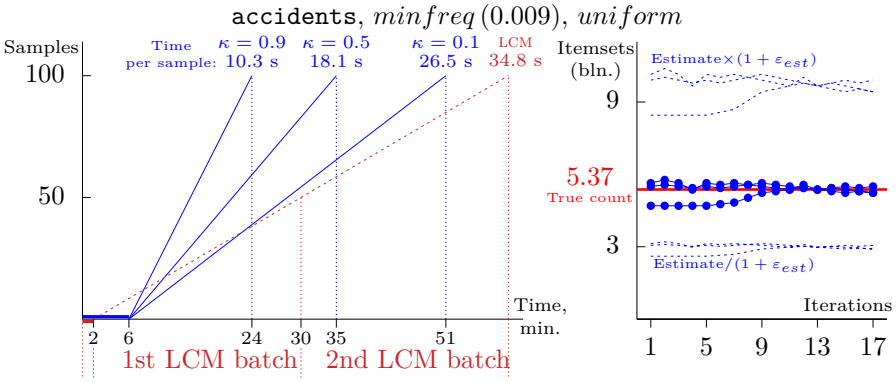
A characteristic experiment in detail We use the `accidents` dataset (469 items, 340 183 transactions) and $\theta = 0.009$ (3000 transactions), which results in a staggering number of 5.37 billion frequent itemsets. We run WeightGen with values of $\kappa \in \{0.1, 0.5, 0.9\}$. (Note that the estimation phase is identical for all three cases.) The baseline sampler is constructed as follows. lcm is first run in counting mode, which only returns the total number of itemsets. Then, for each batch, 50 random line numbers are drawn, and the corresponding itemsets are printed while lcm is enumerating the solutions[10]. The latter phase is implemented with the standard Unix utility '`awk`'.

Figure 4.6 illustrates the results. The counting mode of lcm is roughly 4.5 minutes faster than the estimation phase of EFlexics. Generating samples

[8] `http://research.nii.ac.jp/~uno/codes.htm`, ver. 3

[9] `http://fimi.ua.ac.be/data/`

[10] Storing all itemsets on disk provides no benefits: it increases the mining runtime to 23 minutes and results in a file of 215Gb; simply counting its lines with '`wc -l`' takes 25 minutes.



a) Sampling runtime comparison b) Estimation accuracy

Figure 4.6: a) EFLEXICS generates two batches of 50 samples faster than a sampler derived from LCM, regardless of error tolerance. b) EFLEXICS with the *uniform* quality converges to a high-quality estimate of the total number of itemsets in a small number of iterations (three different random seeds shown). Practical error of the estimation phase is substantially lower than theoretical bounds, which indirectly signals high sampling accuracy.

from the output of LCM, on the other hand, is considerably slower: it takes approximately 35s to sample one itemset, whereas EFLEXICS takes from 10s to 27s per sample, depending on error tolerance κ. As a result, EFLEXICS samples two batches faster than LCM regardless of its parameter values. Moreover, with $\kappa = 0.9$ it samples all 100 itemsets even before the first batch is returned by LCM.

Thus, the proposed sampler outperforms a sampler derived from an efficient itemset miner, even though the experimental setup favors the latter. First, non-uniform weighted sampling would require more advanced computations with itemsets, which would increase the costs of both counting and sampling with LCM. Second, EFLEXICS could also benefit from the exact count obtained by LCM and start sampling after 1.5 minutes. Third, the individual itemsets sampled from the output of an algorithm based on deterministic search are not *exchangeable*. Figure 4.7 illustrates this: due to LCM's search order, certain items only occur at the beginning of batches, while for EFLEXICS, the order within a batch is random.

The accuracy of FLEXICS in this scenario can be evaluated indirectly, by comparing the estimate of the total number of itemsets obtained at the estimation phase with the actual number. The error tolerance of the estimation phase is $\varepsilon_{est} = 0.8$ (see Section 4.4 for details). Figure 4.6b demonstrates

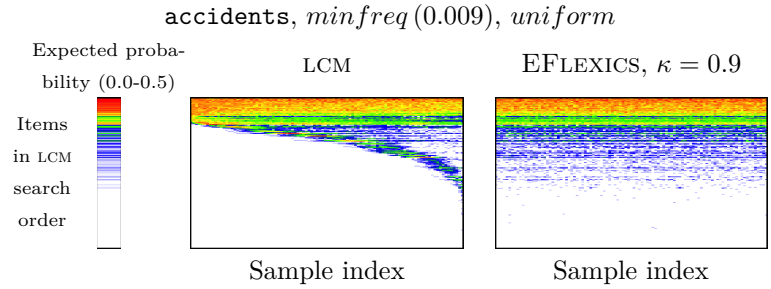


Figure 4.7: The probability of observing a given item at a certain position in a batch by EFLEXICS is close to the expected probability of observing this item in a random itemset, which indicates high sampling accuracy. The samples by the LCM-based sampler are not exchangeable, i.e., certain items are under- or oversampled at certain positions in a batch, depending on their position in LCM's search order.

	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	Density	θ	Itemsets, bln.	Counting, min		Sampling, s	
						LCM	EFLEXICS	LCM	EFLEXICS
`accidents`	340 183	469	7%	0.01	5.4	1.5	6.5	33.8	10.3
`connect`	67 557	130	33%	0.18	16.9	0.01	0.4	59.0	0.4
`kosarak`	990 002	41 271	0.02%	0.04	10.9	4.9	456.3	73.0	294.9
`pumsb`	49 046	7117	1%	0.15	1.1	0.1	1.2	18.1	0.8

Table 4.10: EFLEXICS generates individual samples considerably faster than LCM, although it is slower in counting. The `kosarak` dataset poses a significant challenge to EFLEXICS due to its number of items and sparsity that complicate the propagation of XOR constraints.

that, in practice, the error is substantially lower than the theoretical bound. Furthermore, 3 to 9 iterations suffice to obtain an accurate estimate. Similar to previous experiments, accurate input from the estimation phase alleviates theoretical risks and is expected to enable accurate sampling.

Table 4.10 summarizes the results. On three out of four datasets, LCM is faster in counting itemsets, but considerably slower in generating individual samples, which is even more pronounced on `connect` and `pumsb` than on `accidents`. The results are opposite on the `kosarak` dataset, which is in line with the theoretical expectations (see Section 4.5): the large number of items and the sparsity of the dataset sharply increase the costs of XOR constraint propagation. As a result, enumeration with ECLAT within EFLEXICS becomes considerably slower

than with LCM (augmenting LCM to handle XOR constraints might provide a solution, but is challenging from an implementation perspective).

Q4: Pattern set sampling In order to demonstrate the flexibility of our approach and the promised benefits of weighted constrained pattern sampling, i.e., 1) diversity and quality of results, 2) utility of constraints, and 3) the potential for anytime exploration, we here address the problem of sampling non-overlapping 2-tilings as introduced in Section 4.7. We re-use the implementation of GFLEXICS from the itemset sampling experiments, only modifying the declarative specification of the CSP. Constraints on individual patterns are identical to previous experiments: $minfreq(\theta) \wedge closed \wedge minlen(5)$.

Table 4.11 shows parameters and runtimes for sampling 2-tilings proportional to *area*. The time to sample a single 2-tiling is suitable for pattern-based data exploration, where tilings are inspected by a human user, as it exceeds 5s only on the `german` dataset. For several settings, the estimation phase runtime slightly exceeds the runtime of enumerating all solutions. However, for the settings with a large number of pattern sets, which are arguably the primary target of pattern samplers, the opposite is true. For example, in the `vote` experiment with 170 million tilings, the estimation phase runtime only amounts to 8% of the complete enumeration runtime, which demonstrates the benefits of the proposed approach.

The upper part of Figure 4.8 shows six random 2-tilings sampled from the `vote` dataset. Constraints ensure that the individual tiles comprising each 2-tiling do not overlap, simplifying interpretation. Moreover, the *set* of tilings is diverse,

					CP4IM	GFLEXICS	
	θ	λ	Tilt bound $\hat{r}$	Tilings, mln.	Enumeration, min	Estimation, min	Per sample, s
`german`	0.22	3	25.4	11.2	8.2	12.6	15.3
`heart`	0.30	5	13.3	2.2	1.0	3.3	3.9
`hepatitis`	0.26	5	12.4	7.2	1.9	2.6	3.6
`kr-vs-kp`	0.31	4	13.1	20.3	18.5	3.5	5.1
`primary`	0.03	5	50.3	24.9	5.5	4.0	4.5
`vote`	0.10	5	15.3	170.1	37.0	2.9	4.4

Table 4.11: In general, it takes approximately 4s to sample a 2-tiling, which is suitable for anytime exploration. Runtime benefits of the sampling procedure are the largest for the settings with the largest tiling counts (`kr-vs-kp`, `primary`, and `vote`).

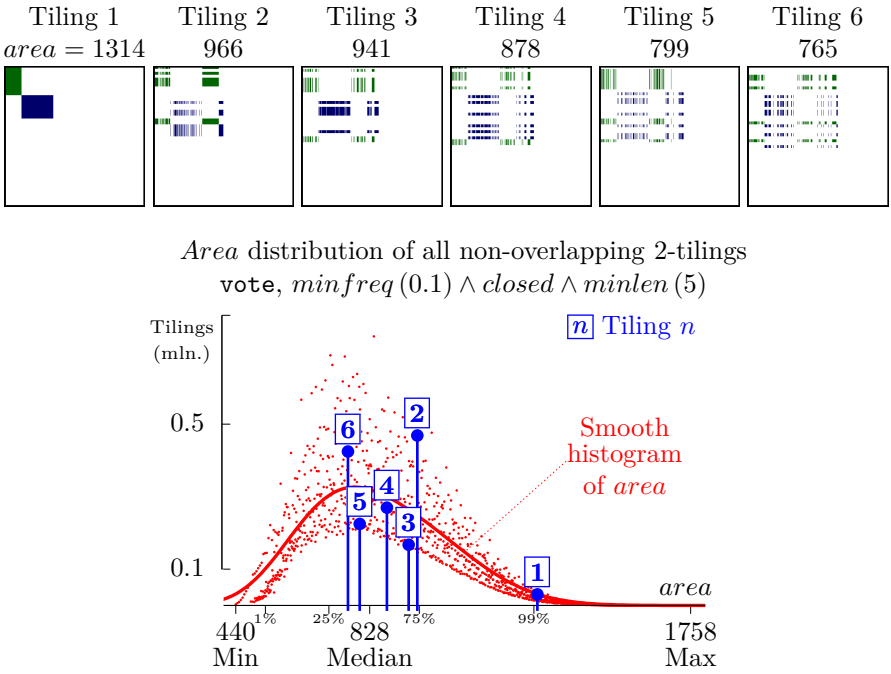


Figure 4.8: **Top**: Six 2-tilings sampled consecutively from the `vote` dataset. The tilings are diverse, i.e., cover different regions in the data, a property essential for pattern-based data exploration. (Note that while the sampled tilings are fair random draws, the images are not random: the tilings were sorted by *area* descending, and items and transactions were re-arranged so that the cells covered by tilings with larger *area* are as close to each other as possible.) **Bottom**: Qualities (*area*) of the samples, indicated by vertical bars, tend towards a dense region between the 25th and the 75th percentile.

i.e., the tilings are dissimilar to each other. They cover different regions in the data, revealing alternative structural regularities.

The lower part of Figure 4.8 shows the *area* distribution of all 2-tilings that satisfy the constraints, obtained by complete enumeration. Qualities of 5 out of 6 tilings fall in the dense region between the 25th and 75th percentile, indicating high sampling accuracy. This is completely expected from the problem statement. In practice, pattern quality measures, like *area*, are only an approximation of application-specific pattern interestingness, thus diversity of results is a desirable characteristic of a pattern sampler as long as the quality of individual patterns is sufficiently high. To sample patterns from the right tail (i.e., with exceptionally

high qualities) more frequently, the sampling task could be changed, e.g., either by choosing another sampling distribution or by enforcing constraints on *area*.

4.9 Discussion

The experiments demonstrate that Flexics delivers the promised benefits: 1) it is flexible in that it supports a wide range of pattern constraints and sampling distributions in itemset mining as well as the novel pattern set sampling task; 2) it is anytime in that the time it takes to generate random patterns is suitable for online data exploration, including the settings with large datasets or large solution spaces; and 3) by virtue of high sampling accuracy in all supported settings, sampled patterns are diverse, i.e., originate from different regions in the solution space. The theoretical guarantees ensure that the empirical observations extend reliably beyond the studied settings. Furthermore, practical accuracy is substantially higher than theory guarantees. The results confirm that pattern mining can benefit from the latest advances in AI, particularly in weighted constrained sampling for SAT. In this section, we discuss potential applications, advantages, and limitations of the proposed approach.

The primary application of pattern sampling involves showing sampled patterns directly to the user. In exploratory data analysis, the mining task is often ill-defined, i.e., the quality measure and the constraints reflect the application-specific pattern interestingness only approximately [34]. Pattern sampling allows obtaining diverse and representative sets of patterns in an anytime manner. Owing to its flexibility, Flexics allows experimenting with various task formulations using the same algorithm.

Furthermore, the theoretical guarantees enable applications beyond displaying the sampled patterns: Flexics can be plugged into algorithms that use patterns as building blocks for pattern-based models, yielding anytime versions thereof with (ε, δ)-approximation guarantees of their own derived from Flexics' guarantees. Example approaches include community detection with Eclat [10] or outlier detection with two-step sampling [61]. The authors note that the formulation of the mining task has a strong influence on the results in the respective applications. Flexics allows the algorithm designer to experiment with these choices and thus to obtain variants of these approaches, perhaps with better application performance.

The flexibility also provides algorithmic advantages. In addition to being agnostic of the quality measure φ and the constraint set $\mathcal{C}$, Flexics is also agnostic of the underlying solution space and the oracle, as long as 1) solutions can be encoded with binary variables and 2) the oracle supports

XOR constraints. Thus, FLEXICS provides a principled method to convert a pattern enumeration algorithm into a sampling algorithm, which amounts to implementing the mechanism to handle XOR constraints. This allows re-using algorithmic advances in pattern mining for developing pattern samplers, which we accomplished with CP4IM and ECLAT.

Most importantly, FLEXICS' black-box nature simplifies extensions to new pattern languages. For example, possible extensions of GFLEXICS cover a variety of pattern set languages in [69], e.g., conceptual clustering. EFLEXICS can be extended to sample other binary pattern languages, e.g., association rules [3] or redescriptions [118]. In contrast, MCMC algorithms, like LRW, are based on local neighborhood enumeration, which is uncommon in traditional pattern mining techniques, and thus require distinctive design and implementation principles for novel problems.

On the other hand, FLEXICS only supports pattern languages that can be compactly represented with binary variables, such as the itemsets and pattern sets studied in this chapter. This essentially limits it to propositional discrete (binary, categorical, or discretized numeric) data. While in principle structured pattern languages, e.g., sequences or graphs, could also be modeled using this framework, the number of variables would rise sharply, which would negatively affect performance. Devising hashing-based sampling algorithms for non-binary domains is an open problem. In particular, sequence mining can be encoded with integer variables [79]; generalized XOR constraints [65] is one possible research direction. Alternatively, as the M4RI library [96] that we base our implementation on is optimized for *dense* $\mathbb{F}_2$ matrices, certain performance issues may be addressed with Gaussian elimination and echelonization algorithms optimized for sparse matrices [21].

Another limitation concerns the bounded tilt assumption regarding sampling distributions: many common quality measures, e.g., χ^2, *information gain* [106], or *weighted relative accuracy* [93], have high or even effectively infinite tilts (if φ can be arbitrarily close to 0). Such quality measures could be tackled with divide-and-conquer approaches [36, Section 6] or alternative estimation techniques [52]. This requires the capacity to efficiently handle constraints of the form $a \leq \varphi(p) \leq b$, which is possible for a number of quality measures, including the ones listed above.

4.10 Conclusion

We proposed FLEXICS, a flexible pattern sampler with theoretical guarantees regarding sampling accuracy. We leveraged the perspective on pattern mining

as a constraint satisfaction problem and developed the first pattern sampling algorithm that builds upon the latest advances in sampling solutions in SAT. Experiments show that FLEXICS delivers the promised benefits regarding flexibility, efficiency, and sampling accuracy in itemset mining as well as in the novel task of pattern set sampling and that it is competitive with state-of-the-art alternatives. In Chapter 5, we exploit the strengths of FLEXICS in the interactive setting.

Directions for future work include extensions to richer pattern languages and relaxing assumptions regarding sampling distributions (see Section 4.9 for a discussion). Specializing the sampling procedure towards typical mining scenarios may allow for deriving tighter theoretical bounds and improving the practical performance; examples include specific constraint types (e.g., anti-/monotone), shapes of sampling distributions (e.g., right-peaked distributions, similar to Figure 4.8), and iterative mining. Following the future developments in weighted constrained sampling in AI may provide insights for improving various aspects of FLEXICS or pattern sampling in general.

Chapter 5

Interactive pattern sampling

5.1 Introduction

Our overarching aim is to enable analysts—such as the ones described in the basketball scenario in Section 1.1—to discover small sets of patterns from data that they consider interesting. This translates to the following three specific requirements. First, we require our approach to yield *concise and diverse result sets*, effectively avoiding the pattern explosion. Second, our method should *take the user's interests into account* and ensure that the results are relevant. Third, it should achieve this *with limited effort on behalf of the user*.

We investigated tools for satisfying these requirements in the previous chapters: we addressed the first requirement with pattern sampling in Chapter 4 and the last two with active preference learning in Chapter 3. However, neither approach satisfies all three requirements: Flexics, the proposed pattern sampler, is agnostic of the underlying sampling task, whereas APLe, the proposed method to learn a model of user's interests from her feedback, does not explicitly focus on conciseness and diversity, as these are expected to be ensured by the external search component.

To address these shortcomings, we propose an approach that combines pattern sampling with interactive data mining techniques and thus satisfies all three

This chapter is based on the conference paper "Learning what matters – Sampling interesting patterns" [47].

requirements. In particular, we introduce the LetSIP algorithm, for **Le**arn **t**o **S**ample **I**nteresting **P**atterns, which implements the steps the *Mine, Interact, Learn, Repeat* framework with the approaches proposed in this thesis. It samples a small set of patterns with Flexics, receives feedback from the user, exploits the feedback to learn new parameters for the sampling distribution with preference learning similar to APLe, and repeats these steps. As a result, the user may utilize a compact diverse set of interesting patterns at any moment, blurring the boundaries between learning and discovery modes.

We satisfy the first requirement by using a sampling technique that samples high quality patterns with high probability. While sampling does not guarantee diversity *per se*, we demonstrate that it gives concise yet diverse results in practice. Moreover, sampling has the advantage that it is *anytime*, i.e., the result set can grow by user's request. LetSIP's sampling component is based on Flexics (Chapter 4).

The second requirement is satisfied by learning what matters to the user, i.e., by interactively learning the distribution patterns are sampled from. This allows the user to steer the sampler towards subjectively interesting regions. LetSIP builds upon APLe (Chapter 3) in that it also uses *preference learning* to learn to rank patterns and sampling to randomize query selection.

Although user effort can partially be quantified by the total amount of input that needs to be given during the analysis, the third requirement also concerns the time that is needed to find the first interesting results. For this it is of particular interest to study the *trade-off between exploration and exploitation*, i.e., eliciting feedback about various pattern space regions vs. showing high-scoring patterns according to the current approximation of subjective quality. As mentioned, one of the benefits of interactive pattern sampling is that the boundaries between learning and discovery are blurred, meaning that the system keeps on learning while it continuously aims to discover potentially interesting patterns.

We evaluate the performance of the proposed algorithm and compare it to the state-of-the-art in interactive pattern mining by emulating the interests of a user. The results confirm that the proposed algorithm has the capacity to learn what matters based on little feedback from the user. More importantly, the LetSIP algorithm demonstrates favorable trade-offs concerning both quality–diversity and exploitation–exploration when compared to existing methods.

This chapter is organized as follows. We review related work in Section 5.2. In Section 5.3, we present the major ingredients of LetSIP. The experimental evaluation in Section 5.4 illustrates the steps of the algorithm, investigates the effect of the parameters on its performance, and compare it with state-of-the-art alternatives. Finally, we present our conclusions in Section 5.5.

5.2 Related work

Several classes of related work are discussed in the previous chapters, in particular interactive pattern mining and preference learning in Chapter 3 and constraint-based mining and pattern sampling in Chapter 4. Hence, in this chapter we focus on the anytime aspect of the *Mine* step implementations by several algorithms described in Section 3.2, some of which are based on pattern sampling. This includes OCM, PRIIME, and IPM.

With regard to the *Learn* step, OCM [18] learns to rank patterns according to user's interests. However, it does not use the learned model during the *Mine* step, but rather learns to allocate computational time to various mining algorithms: the higher the patterns returned by an algorithm are ranked, the more computational time is allocated to it. This scheme is favorable for sampling algorithms: if the sampling distribution matches the user's interests, the more time a sampler gets, the more likely it is that it samples subjectively interesting patterns. In particular, OCM uses two-step samplers with different sampling distributions [19]. However, the two-step samplers only support a limited number of distributions derived from *objective* quality measures. Combined with the lack of support for constraints, this potentially leads to the issues similar to the ones demonstrated in Section 4.8: many of the generated samples would not be interesting to the user, e.g., due to being infrequent or originating from a distribution that does not correspond to subjective preferences.

PRIIME [13] obtains the anytime property by treating an output of the (black-box) mining algorithm as a stream of candidate patterns, which arrive in sequential batches and from which small subsets are used to query the user for feedback. It uses graded feedback (i.e., ratings) to learn a pattern scoring function from the feedback provided by the user, which is a linear combination of pattern features and feature weights. Query selection is guided by the potential impact on the feature weights. This procedure results in two major drawbacks with respect to anytime mining. First, processing the output of a deterministic mining algorithm sequentially has a negative influence on the diversity of consecutive queries (cf. Figure 4.7) and thus potentially hinders learning as the information about ratings of patterns in the early batches may not be relevant for subsequent batches. Second, PRIIME is not genuinely anytime, as it is split into two distinct phases: *Learning* and *Discovery*. The learning phase is biased towards exploration as the patterns that PRIIME shows to the user are aimed to be informative for the learner, rather than interesting to the user. Thus, the exploitation is essentially postponed to the discovery phase. Even then, patterns are processed in the enumeration order imposed by the underlying mining algorithm, which implies that certain interesting patterns can only be returned at the end of the analysis session.

Algorithm 5 LetSIP

Input: Dataset $\mathcal{D}$, minimal frequency threshold θ
Output: Query size k, query retention l, range A, cell sampling strategy ς
SCD: regularization parameter λ, iterations T; Flexics: error tolerance κ

▷ *Initialization*
Ranking function $h_0 = \textsc{Logistic}(\vec{0}, A)$ ▷ Zero weights lead to uniform sampling
Feedback $U \leftarrow \emptyset$, $Q_0^* \leftarrow \emptyset$
▷ *Mine*, *Interact*, *Learn*, *Repeat* loop
for $t = 1, 2, \ldots$ **do**
  $R = \textsc{TakeFirst}(Q_{t-1}^*, l)$ ▷ Retain top patterns from the previous iteration
  Query $Q_t \leftarrow R \cup \textsc{SamplePatterns}(h_{t-1}) \times (k - |R|)$ times
  $Q_t^* = \textsc{Order}(Q_t)$, $U \leftarrow U \cup Q_t^*$ ▷ Ask user to order patterns in Q_t
  $h_t \leftarrow \textsc{Logistic}(\textsc{LearnWeights}(U; \lambda, T), A)$

function SamplePatterns(Sampling weight function $\omega : \mathcal{L} \to [A,\ 1]$)
  $C = \textsc{FlexicsRandomCell}(\mathcal{D},\ freq(\cdot) \geq \theta, \omega; \kappa)$
  if $\varsigma = \textsc{Top}(m)$ **then return** m highest-weighted patterns
  else if $\varsigma = \textsc{Random}$ **then return** $\textsc{PerfectSample}(C, \omega)$

To the best of our knowledge, IPM [12] is the only existing approach to interactive itemset sampling that directly learns the parameters of the sampling distribution. Its sampling component is based on the LRW sampler [72], which we discussed in Chapter 4. IPM uses binary feedback ("likes" and "dislikes") to update weights of individual items. Itemsets are sampled proportional to the product of weights of constituent items. Thus, the model of user interests in IPM is fairly restricted. Moreover, it potentially suffers from convergence issues typical for MCMC. We empirically compare LetSIP with IPM in Section 5.4.

5.3 Algorithm

Key questions concerning instantiations of the *Mine, interact, learn, repeat* framework include 1) the feedback format, 2) learning quality measures from feedback, 3) mining with learned measures, and crucially, 4) selecting the patterns to show to the user. As pattern sampling has been shown to be effective in mining and learning, we present LetSIP, a sampling-based instantiation of the framework, which employs Flexics, and apply it to the itemset mining task; see Chapters 2 and 4 for definitions. The sequel describes the components of LetSIP. Algorithm 5 shows its pseudocode.

Mining patterns by sampling Recall that the main goal is to discover patterns that are subjectively interesting to a particular user. We use parameterized logistic functions to measure the interestingness/quality of a given pattern p:

$$\varphi_{\text{logistic}}(p; w, A) = A + \frac{1 - A}{1 + e^{-\vec{w} \cdot \vec{p}}}$$

where $\vec{p}$ is the vector of pattern features for p, $\vec{w}$ are feature weights, and A is a parameter that controls the range of the interestingness measure, i.e. $\varphi_{\text{logistic}} \in (A, 1)$. Examples of itemset features include $Length(p) = |p|/|\mathcal{I}|$, $Frequency(p) = freq(p)/|\mathcal{D}|$, $Items(i, p) = [i \in p]$; and $Transactions(t, p) = [p \subseteq t]$[1], where $[\cdot]$ denotes the Iverson bracket. Weights reflect feature contributions to pattern interestingness, e.g., a user might be interested in combinations of particular items or disinterested in particular transactions. The set of features would typically be chosen by the mining system designer rather than by the user herself. We empirically evaluate several feature combinations in Section 5.4.

Specifying feature weights manually is tedious and opaque, if at all possible. Below we present an algorithm that learns the weights based on easy-to-provide feedback with respect to patterns. This motivates our choice of logistic functions: they enable efficient learning. Furthermore, their bounded range $[A, 1]$ yields distributions that allow efficient sampling directly proportional to $\varphi_{\text{logistic}}$ with FLEXICS. Parameter A essentially controls the *tilt* of the distribution (see Section 4.4).

In this chapter we focus on the learning aspects of interactive pattern sampling, i.e., object ranking and active learning. Therefore, for the sake of brevity, compared to Chapter 4, we restrict ourselves to sampling frequent patterns (i.e., $\mathcal{C} = \{minfreq\}$) and use EFLEXICS as the sampler. However, in principle, LETSIP is agnostic of constraints and thus can be used with any other pattern constraints, e.g., *closed* or *minlen*. The constraint set would also be chosen by the mining system designer.

User interaction & learning from feedback Similar to Chapter 3, we use *ordered feedback*, where a user is asked to provide a total order over a (small) number of patterns according to their subjective interestingness. We assume that there exists an unknown, user-specific target ranking R^*, i.e., a total order over $\mathcal{L}$. The inductive bias is that there exists $\vec{w}^*$ such that $p \succ q \Rightarrow \varphi_{\text{logistic}}(p, \vec{w}^*) > \varphi_{\text{logistic}}(q, \vec{w}^*)$.

In Chapter 3, we showed that pattern *ranking* functions learned with RANKSVM [76] achieve high ranking performance. However, the values of these functions

[1] The last two feature sets are referred to as *Attributes* and *Cover* in Chapter 3.

are only meaningful in the ranking context; for example, see Table 3.5, where all values are negative. This prevents directly plugging the ranking functions learned with RankSVM into a sampling algorithm as a quality measure, which provides additional motivation for learning logistic quality measures and requires us to design a dedicated learning algorithm.

We adopt a problem formulation comparable to RankSVM (see Section 3.3) and use Stochastic Coordinate Descent (SCD) [124], which performed well in experiments in Chapter 3, for minimizing ℓ_1-regularized logistic loss. SCD is an anytime convex optimization algorithm, which makes it suitable for the interactive setting. Its runtime scales linearly with the number of training pairs and the dimensionality of feature vectors. It has two parameters: 1) the number of weight updates (per iteration of LetSIP) T and 2) the regularization parameter λ. However, direct learning of $\varphi_{\text{logistic}}$ is infeasible, as it results in a non-convex loss function. We therefore use SCD to optimize the standard logistic loss, which is convex, and use the learned weights $\vec{w}$ in $\varphi_{\text{logistic}}$. SCD is also used in OCM; however, unlike OCM, we directly use the learned functions for sampling.

Selecting patterns to show to the user An interactive system seeks to ensure faster learning of accurate models by targeted selection of patterns to show to the user; this is known as *active learning* or *query selection*. Randomized methods have been successfully applied to this task in Chapter 3. Furthermore, in large pattern spaces the probability that two redundant patterns are sampled in one (small) batch is typically low. Therefore, a sampler, which produces independent samples, typically ensures diversity within batches and thus sufficient *exploration*. We directly show k patterns sampled by Flexics proportional to $\varphi_{\text{logistic}}$ to the user, for which she has to provide a total order as feedback.

We propose two modifications to Flexics, which aim at emphasizing *exploitation*, i.e., biasing sampling towards higher-quality patterns. First, we employ alternative cell sampling strategies. Normally Flexics draws a perfect weighted random sample, once it obtains a suitable cell. We denote this strategy as $\varsigma = \textsc{Random}$. We propose an alternative strategy $\varsigma = \textsc{Top}(m)$, which picks the m highest-quality patterns from a cell (Line 10 in Algorithm 5). We hypothesize that, owing to the properties of random XOR constraints, patterns in a cell as well as in consecutive cells are sufficiently diverse and thus the modified cell sampling does not disrupt exploration.

Rigorous analysis of (unweighted) uniform sampling by Chakraborty et al. [35] shows that re-using samples from a cell still ensures broad coverage of the solution space, i.e., diversity of samples. Although as a downside, consecutive samples are not i.i.d., the effects are bounded in theory and inconsequential

in practice. We use these results to take license to modify the theoretically motivated cell sampling procedure. Although we do not present a similar theoretical analysis of our modifications, we evaluate them empirically.

Second, we propose to retain the top l patterns from the previous query and only sample $k - l$ new patterns (Lines 4–5). This should help users to relate the queries to each other and possibly exploit the structure in the pattern space.

5.4 Experiments

The experimental evaluation focuses on 1) the accuracy of the learned user models and 2) the effectiveness of learning and sampling. The research questions are as follows:

Q1 *What effect do* LetSIP*'s parameters have on the quality and diversity of sampled patterns?*

Q2 *How does* LetSIP *compare to state-of-the-art approaches?*

Evaluation methodology

In order to perform extensive evaluation, we use the same protocol as in Chapter 3: we emulate users using (hidden) interest models, which the algorithm is supposed to learn from ordered feedback only. More specifically, we assume that R^* is derived from a quality measure φ, i.e., $p \succ q \Leftrightarrow \varphi(p) > \varphi(q)$. Thus, the task is to learn to sample frequent patterns proportional to φ from (short) sample rankings. As φ, we use frequency $freq$, surprisingness $surp$, and discriminativity in labeled data as measured by χ^2 (see Chapter 2 for definitions).

We investigate the performance of the algorithm on ten datasets[2]. For each dataset, we set the minimal support threshold such that there are at least 140 000 frequent patterns. Table 5.1 shows dataset statistics. Each experiment involves 30 iterations (queries). We use the default values suggested by the authors of SCD and Flexics for the auxiliary parameters of LetSIP: $\lambda = 0.001$, $T = 1000$, and $\kappa = 0.9$.

One of the promised benefits of pattern sampling is the diversity of returned pattern collections. *Joint entropy* [141] is a common measure of diversity. Given an (arbitrarily ordered) pattern set $\mathcal{P}$ of size k, joint entropy essentially

[2]Source: `https://dtai.cs.kuleuven.be/CP4IM/datasets/`

	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{D}\rvert$	θ	Number of frequent patterns
anneal	93	812	660	149 331
australian	125	653	300	141 551
german	112	1000	300	161 858
heart	95	296	115	153 214
hepatitis	68	137	48	148 289
lymph	68	148	48	146 969
primary	31	336	16	162 296
soybean	50	630	28	143 519
vote	48	435	25	142 095
zoo	36	101	10	151 806

Table 5.1: Datasets used in experiments

quantifies the overlap of sets of transactions, in which the patterns in $\mathcal{P}$ occur. It is formally defined as follows. Let $[\cdot]$ denote the Iverson bracket, $b' \in \{0,1\}^k$ a binary k-tuple, and $P(b') = \frac{1}{|\mathcal{D}|} \sum_{t \in \mathcal{D}} \prod_{i \in [1,\ k]} [b'_i = 1 \Leftrightarrow \mathcal{P}_i \subseteq t]$ the fraction of transactions in $\mathcal{D}$ covered only by patterns in $\mathcal{P}$ that correspond to non-zero elements of b' (e.g., if $k = 3$ and $b' = 101$, we only count the transactions covered by the 1st and the 3rd pattern and *not* covered by the 2nd pattern). Joint entropy H_J is defined as $H_J(\mathcal{P}) = -\sum_{b \in \{0,1\}^k} P(b) \times \log_2 P(b)$. H_J is measured in bits and bounded from above by k. The higher the joint entropy, the more diverse are the patterns in $\mathcal{P}$ in terms of their occurrences in $\mathcal{D}$.

We evaluate performance using *cumulative regret*, which is the difference between the ideal value of a certain measure M and its observed value, summed over iterations. Regret takes both accuracy and the rate of learning into account. We use the maximal and average quality φ in a query and joint entropy as performance measures. To allow comparison across datasets and target measures φ, we use percentile ranks by φ as a non-parametric measure of pattern quality. We divide joint entropy by k: thus, the ideal value of each measure is 1 (e.g., the highest possible φ over all patterns has the percentile rank of 1), and the regret is defined as $\sum 1 - M(Q_i^*)$, where $M \in \{\varphi_{avg}, \varphi_{max}, H_J\}$. We repeat each experiment ten times with different random seeds and report average regret.

	Iteration 1			Iteration 2			...	Iteration 30		
	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,3}$	$p_{2,2}$	$p_{2,3}$		$p_{29,1}$	$p_{30,2}$	$p_{30,3}$
$freq$, $\lvert p\rvert$, ...	$52, 6$	$49, 7$	$48, 9$	$48, 9$	$53, 7$	$54, 9$		$73, 8$	$60, 8$	$54, 8$
Feedback U	$p_{1,3} \succ p_{1,1} \succ p_{1,2}$			$p_{1,3} \succ p_{2,2} \succ p_{2,3}$				$p_{29,1} \succ p_{30,2} \succ p_{30,3}$		
$\varphi = surp$	0.12	0.04	0.20	0.20	0.11	0.10		0.28	0.26	0.12
(pct.rank)	0.51	0.13	0.84	0.84	0.46	0.41		0.99	0.97	0.51
Regret: Max.φ	$1 - 0.84 = 0.16$			0.16				0.01		

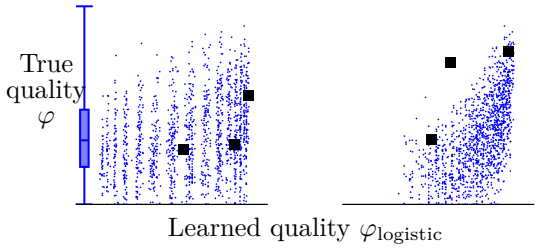


Figure 5.1: We emulate user feedback U using a hidden quality measure φ (here *surp*; the boxplot shows the distribution of φ in the given dataset). The rows above the bar show the properties of the sampled patterns that would be inspected by a user, e.g., *frequency* or *length*, and the emulated feedback. The scatter plots show the relation between φ and the learned model of user interests $\varphi_{\text{logistic}}$ after 1 and 29 iterations of feedback and learning. The performance of the learned model improves considerably as evidenced by higher values of φ of the sampled patterns (squares) and lower regret.

Interactive pattern sampling – an example

Figure 5.1 illustrates the workings of LetSIP and the experimental setup. It uses the `lymph` dataset, the target quality measure $\varphi = surp$, *Items* as features, and the following parameter settings: $k = 3$, $A = 0.1$, $l = 1$, $\varsigma =$ Random.

LetSIP starts by sampling patterns uniformly. A human user would inspect the patterns (items not shown) and their properties, e.g., frequency or length, or visualizations thereof, and rank the patterns by their subjective interestingness; in these experiments, we order them according to their values of φ. The algorithm uses the feedback to update $\varphi_{\text{logistic}}$. At the next iteration, the patterns are sampled from an updated distribution. As $l = 1$, the top-ranked pattern from the previous iteration ($p_{1,3}$) is retained. After a number of iterations, the accuracy of the approximation increases considerably, while the regret decreases. On average, one iteration takes 0.5s on a desktop computer.

We also use this example to illustrate the modifications to cell sampling in

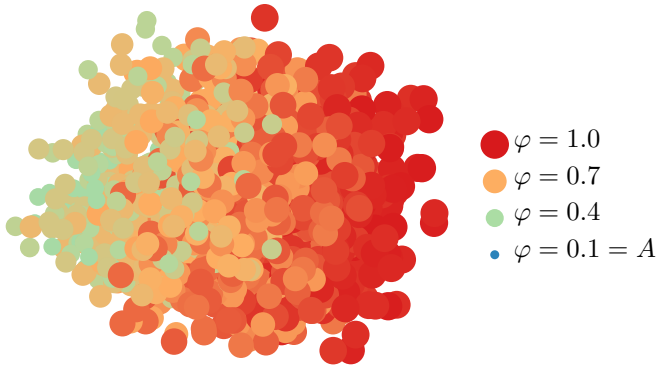


Figure 5.2: The two principal components obtained from the *Items* features of all frequent patterns, i.e., pattern descriptions[4]. The size and the color of a point indicate the value of $\varphi_{\text{logistic}}$ of the corresponding pattern. For clarity, only a 1%-subsample is shown.

FLEXICS that we describe in Section 5.3. Similar to Section 4.6, we visualize the patterns by plotting the two principal components obtained from the description matrix (*Items* features). Figure 5.2 shows all frequent patterns, while Figure 5.3 shows examples of random cells, i.e., the output of FLEXICSRANDOMCELL, from which patterns are chosen by a cell sampling strategy ς.

The cells are different from each other, thus patterns returned from consecutive cells are independent and diverse. In each cell, we highlight the pattern with the highest quality $\varphi_{\text{logistic}}$, which is returned by $\varsigma = \text{TOP}(1)$, along with $P_{\varsigma=\text{RANDOM}}$, the probability that it is *sampled from that cell* if $\varsigma = \text{RANDOM}$. These probabilities do not exceed 0.05, which demonstrates the motivation for alternative cell sampling strategies. As expected, the patterns returned by TOP(1) are concentrated in the regions in the pattern space that are characterized by high values of $\varphi_{\text{logistic}}$. Nevertheless, they are different from each other, thus the diversity across samples is maintained, regardless of the bias towards exploitation.

[4]The PCA coordinates and $\varphi_{\text{logistic}}$ are strongly correlated, because they are computed using the same feature representation for patterns (*Items*).

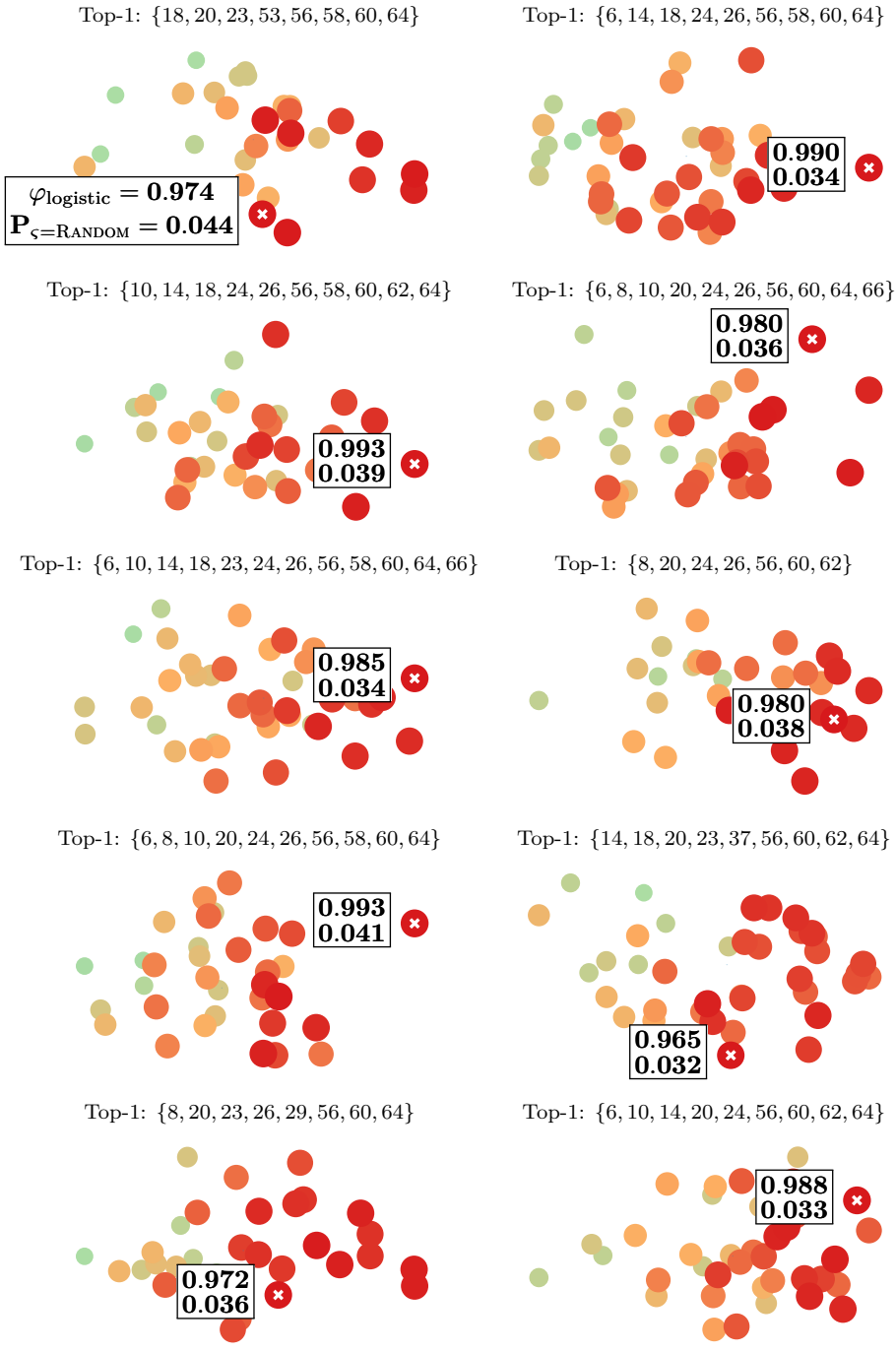


Figure 5.3: Ten random cells obtained with LETSIP. The cells as well as the top patterns within each cell (descriptions shown) are different from each other.

Experimental results

Q1: Evaluating components of LetSIP We investigate the effects of the choice of features and parameter values on the performance of LetSIP, in particular query size k, query retention l, range A, and cell sampling strategy ς. We use the following feature combinations ($\|$ denotes concatenation): *Items* (I); *Items*$\|$*Length*$\|$*Frequency* (ILF); and *Items*$\|$*Length*$\|$*Frequency*$\|$*Transactions* (ILFT). Values for other parameters and aggregated results are shown in Table 5.2.

Increasing the query size decreases the maximal quality regret more than twofold, which indicates that the proposed learning technique is able to identify the properties of target measures from ordered lists of patterns. However, as larger queries also increase the user effort, further we use a more reasonable query size of $k = 5$. Similarly, additional features provide valuable information to the learner. Changing the range A does not affect the performance.

The choice of values for query retention l and the cell sampling strategy allows influencing the exploration-exploitation trade-off. Interestingly, retaining one highest-ranked pattern results in the lowest regret with respect to the *maximal* quality. Fully random queries ($l = 0$) do not enable sufficient exploitation, whereas higher retention ($l \geq 2$)—while ensuring higher *average* quality—prevents exploration necessary for learning accurate weights.

The cell sampling strategy is the only parameter that clearly affects joint entropy, with purely random cell sampling yielding the lowest regret. However, it is also results in the highest quality regrets, which negates the gains in diversity. Taking the best pattern according to $\varphi_{\text{logistic}}$ ensures the lowest quality regrets and joint entropy equivalent to other strategies. Based on these findings, we use the following parameters in the remaining experiments: $k = 5$, features = ILFT, $A = 0.5$, $l = 1$, $\varsigma = \textsc{Top}(1)$.

The largest proportion of LetSIP's runtime costs is associated with sampling (costs of weight learning are low due to a relatively low number of examples). The most important factor is the number of items $|\mathcal{I}|$: the average runtime per iteration ranges from 0.8s for `lymph` to 5.8s for `australian`, which is suitable for online data exploration. See Chapter 4 for more information about the scalability of the sampling component.

Q2: Comparing with alternatives We compare LetSIP with APLe (see Chapter 3) and IPM [12], an MCMC-based interactive sampling framework. For the former, we use query size k and feature representation identical to LetSIP, query selector MMR($\alpha = 0.3$, $\lambda = 0.7$), $C_{\text{RankSVM}} = 0.005$, and 1000 frequent

		Regret w.r.t. Avg.φ	Regret w.r.t. Max.φ	Regret w.r.t. H_J
Query size k	5	6.4 ± 1.0	1.1 ± 0.5	$\mathbf{13.3 \pm 0.9}$
	10	$\mathbf{5.9 \pm 0.6}$	$\mathbf{0.5 \pm 0.2}$	17.4 ± 0.5
All results below are for query size of $k = 5$				
Features	I	8.2 ± 1.0	1.4 ± 0.6	13.6 ± 0.9
	ILF	6.3 ± 1.4	1.2 ± 0.6	13.2 ± 1.0
	ILFT	$\mathbf{4.6 \pm 0.8}$	$\mathbf{0.9 \pm 0.4}$	$\mathbf{13.1 \pm 0.8}$
Range A	0.5	6.4 ± 1.1	1.2 ± 0.5	$\mathbf{13.2 \pm 0.9}$
	0.1	$\mathbf{6.3 \pm 1.0}$	$\mathbf{1.1 \pm 0.5}$	13.4 ± 0.9
Query retention l	0	8.2 ± 1.2	2.5 ± 0.7	13.4 ± 0.7
	1	6.8 ± 1.0	$\mathbf{0.5 \pm 0.3}$	$\mathbf{13.1 \pm 0.7}$
	2	5.6 ± 0.9	0.6 ± 0.4	13.6 ± 1.1
	3	$\mathbf{4.8 \pm 1.0}$	0.8 ± 0.6	13.3 ± 1.2
Cell sampling ς	RANDOM	10.6 ± 0.7	1.9 ± 0.6	$\mathbf{12.2 \pm 0.6}$
	Top(1)	$\mathbf{5.1 \pm 1.1}$	$\mathbf{0.8 \pm 0.5}$	13.7 ± 1.0
	TOP(2)	5.5 ± 1.1	0.9 ± 0.5	13.6 ± 1.0
	TOP(3)	6.0 ± 1.2	1.0 ± 0.5	13.6 ± 1.0

Table 5.2: Effect of LETSIP's parameters on regret w.r.t. three performance measures. Results are aggregated over datasets, quality measures, and other parameters.

patterns sampled uniformly at random and sorted by $freq$ as the source ranking. To compute regret, we use the top-5 frequent patterns according to the learned ranking function.

To emulate binary feedback for IPM based on φ, we use a technique similar to the one used by the authors: we designate a number of items as "interesting" and "like" an itemset, if more than half of its items are "interesting." To select the items, we sort frequent patterns by φ descending and add items from the top-ranked patterns until 15% of all patterns are considered "liked."

As we were not able to obtain the code for IPM, we implemented its sampling component by materializing all frequent patterns and generating perfect samples according to the learned multiplicative distribution. Note that this approach favors IPM, as it eliminates the issues of MCMC convergence. We request 300 samples (the amount of training data roughly equivalent to that of LETSIP), partition them into 30 groups of 10 patterns each, and use the tail 5 patterns

	Regret: avg.φ			Regret: joint entropy H_J		
	$freq$	χ^2	$surp$	$freq$	χ^2	$surp$
LetSIP	2.4 ± 0.5	$\mathbf{2.4 \pm 0.1}$	$\mathbf{4.5 \pm 1.4}$	$\mathbf{11.7 \pm 0.6}$	$\mathbf{11.7 \pm 0.5}$	$\mathbf{15.9 \pm 1.1}$
IPM	15.5 ± 1.8	12.8 ± 2.3	15.5 ± 1.8*	15.7 ± 1.9	15.4 ± 1.9	19.8 ± 2.1*
APLe	$\mathbf{0.0 \pm 0.0}$	4.5 ± 3.8	5.3 ± 3.9	–	–	–

Table 5.3: LetSIP has considerably lower regrets than alternatives w.r.t. quality and, for samplers, diversity as quantified by joint entropy. (For $\varphi = surp$ (marked by *), IPM fails for 7 out of 10 datasets due to double overflow of multiplicative weights.)

in each group for regret calculations. Following the authors' recommendations, we set the learning parameter to $b = 1.75$. For the sampling-based methods LetSIP and IPM, we also report the diversity regret as measured by joint entropy.

Table 5.3 shows the results. The regret of LetSIP is substantially lower than that of either of the alternatives. (It is lower than in Table 5.2, as the specific parameter combination suggested by the previous experiments is used.) The advantage over IPM is due to a more powerful learning mechanism and feature representation. IPM's multiplicative weights are biased towards longer itemsets and items seen at early iterations, which may prevent sufficient exploration, as evidenced by higher joint entropy regret. Non-sampling method APLe performs the best for $\varphi = freq$, which can be represented as a linear function of the features and learned by RankSVM with the linear kernel. It performs substantially worse in other settings and has the highest variance, which reveals the importance of informed source rankings and the cons of pool-based active learning. These results validate the design choices made in LetSIP.

5.5 Conclusion

We presented LetSIP, a sampling-based instantiation of the *Mine, interact, learn, repeat* interactive pattern mining framework. The user is asked to rank small sets of patterns according to their (subjective) interestingness. The learning component uses this feedback to build a model of user interests via active preference learning. The model directly defines the sampling distribution, which assigns higher probabilities to more interesting patterns. The sampling component uses the Flexics sampler, which we modify to facilitate control over the exploration-exploitation balance in active learning.

We empirically demonstrated that LetSIP satisfies the key requirements of an interactive mining system. We applied it to itemset mining, using a well-principled method to emulate a user. The results demonstrate that LetSIP learns to sample diverse sets of interesting patterns. Furthermore, it outperforms two state-of-the-art interactive methods. This confirms that it has the capacity to tackle the pattern explosion while taking user interests into account.

Directions for future work include extending LetSIP to other pattern languages, e.g., association rules, investigating the effect of noisy user feedback on the performance, and formal analysis, e.g., with multi-armed bandits [54]. A user study is necessary to evaluate the practical aspects of the proposed approach.

Chapter 6

Conclusion

The concluding chapter summarizes the contributions of this thesis and discusses open problems and directions for future work.

6.1 Summary and conclusions

We investigated algorithmic approaches to the problem of interactive pattern mining, which are partially inspired by research in information retrieval and machine learning. Our objective was to tackle a variety of issues hampering the practical adoption of pattern mining for exploratory data analysis and to lay the foundations for making it more accessible to non-expert users. The crucial issue was to develop efficient and effective methods to identify and account for (subjective) interests and goals of the particular user in the mining process. This issue was addressed only to a limited extent by previous research in pattern mining , while having been tackled in information retrieval research (*and* practice) rather successfully, which motivated us to explore the connections.

To this end, we framed interactive pattern mining as an interactive learning problem and proposed a high-level framework that can be summarized by the adage "Mine, interact, learn, repeat." Individual contributions concerned separate steps within this framework. First, we proposed to learn subjective pattern interestingness measures from the ordered feedback by means of preference learning and developed active learning heuristics that minimize user effort. Second, we introduced a pattern sampling algorithm that enables anytime generation of patterns while providing accuracy and efficiency guarantees. Third,

we presented an end-to-end implementation of the framework that combines the benefits of interactive preference learning and pattern sampling.

We now review the research questions outlined in the introduction and the answers contributed by this thesis.

Q1 *What properties are essential for a mining algorithm as a part of an interactive mining framework?*

In order to facilitate data exploration by the user and to avoid interfering with her analysis flow, the mining algorithm ought to be *adaptable* and *anytime*. The former property is necessary to obviate the need for complex modeling of the target results of the analysis up-front and to make the algorithm accessible to users who are not experts in pattern mining. The latter property allows the user to explore a large number of directions initially and let the algorithm focus on the interesting ones later on.

We contributed three mining algorithms that satisfy these properties (Table 6.1):

APLe An algorithm that adapts to the user by learning to (re-)rank patterns interactively, from user feedback (Chapter 3).

Flexics A pattern sampling algorithm that obtains the anytime property by replacing exhaustive search for patterns with random generation of patterns according to a pre-specified distribution, which assigns higher probabilities to more interesting patterns: the longer it runs, the more interesting patterns are generated (Chapter 4).

LetSIP An end-to-end interactive pattern mining algorithm that learns the sampling distribution interactively, thus combining the benefits of the previous approaches (Chapter 5).

The adaptability of the learning algorithms is based on learning subjective pattern interestingness measures, which we discuss below.

Method	*Mine*	*Interact*	*Learn*
APLe (Chapter 3)	Top-k mining	Ordered feedback	Preference learning (e.g., RankSVM)
LetSIP (Chapter 5)	Flexics sampler (Chapter 4)	Ordered feedback	Preference learning (SCD)

Table 6.1: Overview of interactive pattern mining methods proposed in this thesis organized according to the "Mine, interact, learn, repeat" framework.

Q2 *What kind of feedback is required to organize a convenient and effective interaction of a user with the mining process?*

We investigated *ordered feedback*, where the user is asked to order, or rank, small sets of patterns from the (subjectively) most interesting to the least interesting (Chapters 3 and 5). This allowed us to turn to the *preference learning* paradigm, which has also been successfully applied in information retrieval. We contributed two learning techniques based on this paradigm and empirically demonstrated that a relatively small amount of pattern rankings suffices to learn quality measures that allow discovering novel, previously unseen interesting patterns.

Q3 *How can the total effort required from the user to identify subjectively interesting patterns be minimized?*

We designed a number of active learning heuristics, including several techniques that are agnostic of the underlying preference learner, and one that is tailored for the RankSVM learner. Although experiments showed a trade-off between accuracy of learned rankings and user effort, alternating between mining and learning performed better than a non-iterative baseline (Chapter 3). Moreover, we proposed and evaluated modifications to Flexics that allow to control the exploration-exploitation trade-off in interactive pattern sampling (Chapter 5).

Q4 *Can existing learning techniques leverage the user feedback to learn an accurate user model?*

Building upon the insights from information retrieval, we were able to apply preference learning techniques in the context of interactive pattern mining. Our contributions cover two facets of learning: *batch learning*, where patterns used to elicit user feedback are mined in advance, and *anytime learning*, where they are synthesized on the fly.

Batch learning We empirically evaluated a number of preference learning algorithms, including parameter tuning, and feature representations for patterns (Chapter 3).

Anytime learning Finally, we developed an anytime method for preference learning in pattern mining, which is based on *stochastic coordinate descent*, an anytime loss minimization technique (Chapter 5).

Q5 *How can the learned user models be used to discover subjectively more interesting patterns?*

Using the preference learning paradigm suggests a dual view on interactive pattern mining: from the conceptual perspective the primary goal is to learn to rank patterns before showing them to the user, while from the technical

perspective it boils down to learning subjective pattern interestingness measures. Based on the latter observation, we contributed two ways to use the learned measures to mine novel patterns:

- We plugged in the ranking functions learned by APLe into a beam search-based top-k mining algorithm, where they were primarily used to order intermediate candidate patterns in the "beams" (Chapter 3).
- In LetSIP, owing to the black-box nature of Flexics, a pattern sampler also proposed in this thesis, we were able to sample patterns directly from a distribution defined by the learned ranking function. Furthermore, this allowed us to blur the lines between the learning and the mining phases, resulting in a self-contained implementation of our framework (Chapter 5).

In both cases, we empirically demonstrated that the learned ranking functions have the capacity to identify novel patterns that are more interesting than the patterns seen during the learning phase.

6.2 Future work

We discuss directions for future research, grouping them into three categories: 1) theoretical aspects, 2) practical aspects, and 3) implications for data-driven applications and decision-making.

Theoretical aspects

Here we list a number of open questions regarding algorithm design and analysis.

Complex data types In this thesis we focused on propositional discrete data, i.e., binary, categorical, or discretized numeric data that are contained in a single table. More complex data types include numeric, sequence, graph, and relational data and require more complex pattern languages, which cannot be compactly represented as feature vectors [13]. This issue affects the two proposed learning techniques as well as Flexics and needs to be resolved to extend the proposed approaches to more complex data types.

Learning pattern-based models We focused on discovering *individual* interesting patterns. A natural next step is to consider learning to construct global pattern-based models that compose multiple patterns and characterize (contextual) dependencies between them. Such models may include patterns

that describe well-known facts *in addition to* novel, subjectively interesting patterns [71]. Together they summarize the dataset and explain the structure in it. In particular, *generative* pattern-based models allow generating synthetic observations as a combination of patterns. This enables advanced forms of analysis, e.g., *what-if analysis* [119], and model validation (see below).

Alternative feedback formats The proposed learning approaches used *explicit* ordered feedback, i.e., assumed that the user feedback is a strict total order over the shown patterns. One of the advantages of the ordered feedback is that it is flexible and can be derived implicitly from a variety of user actions [76]. Alternative feedback formats include *binary feedback* ("like"/"dislike") [46, 12], *graded feedback* (ratings) [13], or combinations thereof [23]. Exploring these options and related learning techniques is an interesting research direction.

Interest drift We assumed that user interests do not change in the course of the analysis session, i.e., that the user's latent subjective pattern ranking is fixed. In practice, intermediate discoveries may affect interests or goals. How to handle this within the interactive loop is an open question.

Formal analysis We presented extensive empirical evidence for the effectiveness of the proposed approaches. Formal analysis of such issues as the convergence of the interactive loop (e.g., using the multi-armed bandits framework [117, 54]) and effects of noisy, imprecise feedback [127] may strengthen the conclusions and provide further insights.

Alternative perspectives We viewed interactive pattern mining as learning subjective pattern rankings and quality measures. Alternative perspectives that are worth exploring include constraint learning, parameter or structure learning for Bayesian models of pattern interestingness, extensions of existing compression-based approaches (e.g., replacing the Minimum Description Length (MDL) principle with the subjective Minimum Message Length (MML) principle [143, 145]), and others.

Practical aspects

The following topics concern issues related to implementation, deployment, and validation of interactive mining systems.

User studies In this thesis we focused on algorithmic aspects of interactive pattern mining, which virtually forced us to resort to user emulation in order to perform extensive experimental evaluation. Undoubtedly, the applicability of the proposed techniques to real-world analysis conducted by human users will need to be validated by separate studies, which would require thorough

experimental design [16]. Assembling reference collections of human judgements regarding patterns is necessary for reproducible experiments, similar to the role served by the TREC collections in information retrieval. Exploration of experiment databases [142] is a possible source of such collections, which can emerge within the KDD research community.

User experience Designing transparent user interfaces is crucial for making pattern mining genuinely useful to non-experts. Important research directions include data visualization, pattern visualization [11], user interaction design, and perhaps explanation of learned user models. (First steps have been made by Boley et al. [18].)

Human preferences One of the key aspects of the proposed approach (and interactive data analysis in general) that requires further exploration and validation is the interpretation of user feedback or, more broadly, the underlying user-specific notion of pattern interestingness. We assumed that 1) the user can order any two patterns, i.e., judge which one of the two is subjectively more interesting, and 2) that there exists a real-valued function of pattern features that governs these judgements. Likewise, the existence of the subjective *utility function* that determines preferences (and decisions) of an individual is the central assumption of the *rational choice theory* [6], a core subfield of economics. These assumptions imply that the preferences are *complete* and *transitive*.

However, they are often violated in practice, as the research in such fields as psychology, behavioral economics, social sciences, and human-computer interaction, has demonstrated empirically. The proposed explanations include limited information processing capabilities of humans (bounded rationality) [62]; cognitive biases [77], e.g., framing effects [135]; contextual, social, and other factors. This results in heuristic preference judgements, which are often formulated on-the-fly (rather than retrieved from a well-defined internal model) and thus do not conform to strict axiomatic models [129].

This poses two challenges regarding practical applications of the proposed approaches: 1) whether user feedback can be translated into training examples for learning algorithms in a straightforward manner and 2) whether learned ranking functions (in particular linear functions investigated in this thesis) can help identify interesting or useful patterns even if the user's notion of interestingness cannot be modeled as a total order.

These challenges connect the two previous items in this section. First, user studies should investigate how human analysts judge the interestingness of patterns and whether this process is affected by the factors outlined above, e.g., framing effects in pattern presentation. The design of interactive data analysis tools should take the findings into account with the purpose of eliciting accurate

and informative feedback either explicitly, as in this thesis, or implicitly, where training data is derived from a more practical feedback format [84, 74].

Second, the effectiveness of diverse lists of high-ranked patterns as the form of output should be validated empirically—in a way validating the extent of the analogy with search engines. (A definitive study should include more flexible ranking functions, e.g., RankSVM with non-linear kernels.) Possible alternative forms of output might explicate contextual relationships between patterns, e.g., by presenting them as a network arranged according to a certain model of subjective interestingness.

Multi-user mode In organizations, multiple users share, at least partially, knowledge and analysis goals. Extending the proposed approaches for interactive pattern-based knowledge management, where models of individual users are mixed with the "organization model", and tackling challenges related to *preference aggregation* [120] are interesting avenues for future research.

Implications for data-driven decision-making

Interactive pattern mining potentially provides users with a powerful and flexible data analysis tool. However, it also burdens the user with the responsibility to avert steering the mining process towards patterns that do not correspond to genuine regularities in the instance space, i.e., to prevent *false discoveries* [102]. This is a difficult task even for well-trained and well-intentioned analysts, due to technical as well as cognitive challenges, e.g., *confirmation bias* [105, 92]. Thus, it is essential to research tools for *pattern validation* that should serve as safeguards against these pitfalls.

Data randomization is a common tool used for constructing such safeguards, as it can be seen as validating the model on external data, even if collecting new data is infeasible. *Swap randomization* techniques [63] construct new data by randomly modifying the original dataset, while preserving its core properties, e.g., row and column sums. Pattern characteristics in the original data are compared to its characteristics in "similar" random data. Generative models allow generating new data directly. *Posterior predictive checks* [58] validate a generative model by comparing the sampled (or "replicated") data with the original data. Both approaches are computationally intensive, therefore integrating them into an interactive loop requires additional research.

Bibliography

[1] C. C. Aggarwal, J. Han, eds. *Frequent pattern mining*. Springer, 2014, p. 471 (cited on pp. 1, 14).

[2] R. Agrawal, T. Imieliński, A. Swami. "Mining association rules between sets of items in large databases". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*. 1993, pp. 207–216 (cited on p. 1).

[3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo. "Fast discovery of association rules". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy. AAAI Press, 1996. Chap. 12, pp. 307–328 (cited on pp. 1, 16, 63, 92).

[4] R. Agrawal, R. Srikant, et al. "Fast algorithms for mining association rules". In: *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*. 1994, pp. 487–499 (cited on p. 1).

[5] N. Ailon. "An active learning algorithm for ranking from pairwise preferences with an almost optimal query complexity". In: *Journal of Machine Learning Research* 13 (2012), pp. 137–164 (cited on pp. 31, 38).

[6] K. J. Arrow. "Economic theory and the hypothesis of rationality". In: *Utility and Probability*. Ed. by J. Eatwell, M. Milgate, P. Newman. Palgrave Macmillan UK, 1990, pp. 25–37 (cited on p. 116).

[7] M. Atzmueller. "Subgroup discovery". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 5.1 (2015), pp. 35–49 (cited on p. 13).

[8] M. Atzmueller, F. Lemmerich. "Fast subgroup discovery for continuous target concepts". In: *Proceedings of the 18th International Symposium on Methodologies for Intelligent Systems (ISMIS '09)*. Springer. 2009, pp. 35–44 (cited on p. 13).

[9] R. Bayardo. “Efficiently mining long patterns from databases”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD ’98)*. 1998, pp. 85–93 (cited on p. 16).

[10] M. Berlingerio, F. Pinelli, F. Calabrese. “ABACUS: Frequent pattern mining-based community discovery in multidimensional networks”. In: *Data Mining and Knowledge Discovery* 27.3 (2013), pp. 294–320 (cited on p. 91).

[11] E. Bertini, D. Lalanne. “Investigating and reflecting on the integration of automatic data analysis and visualization in knowledge discovery”. In: *ACM SIGKDD Explorations Newsletter* 11.2 (2010), pp. 9–18 (cited on p. 116).

[12] M. Bhuiyan, M. A. Hasan. “Interactive knowledge discovery from hidden data through sampling of frequent patterns”. In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 9.4 (2016), pp. 205–229 (cited on pp. 20, 23, 30, 98, 106, 115).

[13] M. Bhuiyan, M. A. Hasan. “PRIIME: A generic framework for interactive personalized interesting pattern discovery”. In: *Proceedings of the IEEE International Conference on Big Data (IEEE Big Data ’16)*. 2016, pp. 606–615 (cited on pp. 20, 23, 31, 97, 114, 115).

[14] M. Boley, T. Gärtner, H. Grosskreutz. “Formal concept sampling for counting and threshold-free local pattern mining”. In: *Proceedings of the 10th SIAM International Conference on Data Mining (SDM ’10)*. 2010, pp. 177–188 (cited on pp. 18, 60, 61, 64).

[15] M. Boley, H. Grosskreutz. “Approximating the number of frequent sets in dense data”. In: *Knowledge and information systems* 21.1 (2009), pp. 65–89 (cited on pp. 18, 60, 61, 64, 83).

[16] M. Boley, M. Krause-Traudes, B. Kang, B. Jacobs. “Creedo — Scalable and repeatable extrinsic evaluation for pattern discovery systems by online user studies”. In: *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA ’15)*. 2015, pp. 20–28 (cited on pp. 20, 23, 116).

[17] M. Boley, C. Lucchese, D. Paurat, T. Gärtner. “Direct local pattern sampling by efficient two-step random procedures”. In: *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’11)*. 2011, pp. 582–590 (cited on pp. 18, 60, 61, 64).

[18] M. Boley, M. Mampaey, B. Kang, P. Tokmakov, S. Wrobel. “One Click Mining — interactive local pattern discovery through implicit preference and performance learning”. In: *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA ’13)*. 2013, pp. 28–36 (cited on pp. 30, 97, 116).

[19] M. Boley, S. Moens, T. Gärtner. "Linear space direct pattern sampling using coupling from the past". In: *Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '12)*. 2012, pp. 69–77 (cited on pp. 18, 60, 61, 64, 83, 97).

[20] F. Bonchi, F. Giannotti, C. Lucchese, S. Orlando, R. Perego, R. Trasarti. "A constraint-based querying system for exploratory pattern discovery". In: *Information Systems* 34.1 (2009), pp. 3–27 (cited on pp. 16, 64).

[21] C. Bouillaguet, C. Delaplace. "Sparse gaussian elimination modulo p: An update". In: *Proceedings of the 18th International Workshop on Computer Algebra in Scientific Computing (CASC '16)*. 2016, pp. 101–116 (cited on p. 92).

[22] J.-F. Boulicaut, A. Bykowski, C. Rigotti. "Free-sets: A condensed representation of boolean data for the approximation of frequency queries". In: *Data Mining and Knowledge Discovery* 7.1 (2003), pp. 5–22 (cited on p. 16).

[23] C. Boutilier, K. Regan, P. Viappiani. "Simultaneous elicitation of preference features and utility". In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*. 2010, pp. 1160–1167 (cited on pp. 31, 115).

[24] S. Brin, R. Motwani, J. D. Ullman, S. Tsur. "Dynamic itemset counting and implication rules for market basket data". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '97)*. 1997, pp. 255–264 (cited on p. 15).

[25] B. Bringmann, S. Nijssen, N. Tatti, J. Vreeken, A. Zimmermann. "Mining sets of patterns". In: *Tutorial at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML/PKDD '10)* (2010) (cited on pp. 17, 59).

[26] B. Bringmann, A. Zimmermann. "One in a million: Picking the right patterns". In: *Knowledge and Information Systems* 18.1 (2009), pp. 61–81 (cited on p. 17).

[27] K. Brinker. "Incorporating diversity in active learning with support vector machines". In: *Proceedings of the 20th International Conference on Machine Learning (ICML '03)*. 2003, pp. 59–66 (cited on pp. 39, 51).

[28] C. Bucilă, J. Gehrke, D. Kifer, W. White. "Dualminer: A dual-pruning algorithm for itemsets with constraints". In: *Data Mining and Knowledge Discovery* 7.3 (2003), pp. 241–272 (cited on pp. 16, 64).

[29] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, G. Hullender. "Learning to rank using gradient descent". In: *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*. 2005, pp. 89–96 (cited on p. 37).

[30] T. Calders, B. Goethals. “Mining all non-derivable frequent itemsets”. In: *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD ’02)*. 2002, pp. 74–86 (cited on p. 16).

[31] T. Calders, C. Rigotti, J.-F. Boulicaut. “A survey on condensed representations for frequent sets”. In: *Constraint-Based Mining and Inductive Databases*. Ed. by J.-F. Boulicaut, L. De Raedt, H. Mannila. Springer, 2006, pp. 64–80 (cited on pp. 15, 59).

[32] P. Campigotto, A. Passerini, R. Battiti. “Active learning of combinatorial features for interactive optimization”. In: *Proceedings of the 5th International Learning and Intelligent Optimization Conference (LION V)*. 2011, pp. 336–350 (cited on p. 31).

[33] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, H. Li. “Learning to rank: From pairwise approach to listwise approach”. In: *Proceedings of the 24th International Conference on Machine Learning (ICML ’07)*. 2007, pp. 129–136 (cited on p. 38).

[34] D. R. Carvalho, A. A. Freitas, N. Ebecken. “Evaluating the correlation between objective rule interestingness measures and real human interest”. In: *Proceedings of the 9th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD ’05)*. 2005, pp. 453–461 (cited on pp. 17, 91).

[35] S. Chakraborty, D. J. Fremont, K. S. Meel, S. A. Seshia, M. Y. Vardi. “On parallel scalable uniform SAT witness generation”. In: *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS ’15)*. Vol. 9035. 2015, pp. 304–319 (cited on pp. 67, 74, 100).

[36] S. Chakraborty, D. J. Fremont, K. S. Meel, M. Y. Vardi. “Distribution-aware sampling and weighted model counting for SAT”. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI ’14)*. 2014, pp. 1722–1730 (cited on pp. 60, 66–70, 92).

[37] S. Chakraborty, K. S. Meel, M. Y. Vardi. “A scalable and nearly uniform generator of sat witnesses”. In: *Proceedings of the 25th International Conference on Computer-Aided Verification (CAV ’13)*. 2013, pp. 608–623 (cited on p. 60).

[38] M. Davidian. *Aren’t we data science?* `http://magazine.amstat.org/blog/2013/07/01/datascience/`. (Online; accessed March 21, 2017). 2013 (cited on p. 12).

[39] T. De Bie. “An information theoretic framework for data mining”. In: *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’11)*. 2011, pp. 564–572 (cited on pp. 20, 29).

[40] T. De Bie. “Maximum entropy models and subjective interestingness: An application to tiles in binary databases”. In: *Data Mining and Knowledge Discovery* 23.3 (2010), pp. 407–446 (cited on p. 19).

[41] L. De Raedt, A. Zimmermann. “Constraint-based pattern set mining”. In: *Proceedings of the 7th SIAM International Conference on Data Mining (SDM ’07)*. 2007, pp. 237–248 (cited on pp. 17, 64).

[43] N. Di Blas, M. Mazuran, P. Paolini, E. Quintarelli, L. Tanca. “Exploratory computing: A comprehensive approach to data sensemaking”. In: *International Journal of Data Science and Analytics* 3.1 (2017), pp. 61–77 (cited on p. 12).

[44] K. Dimitriadou, O. Papaemmanouil, Y. Diao. “Explore-by-example: An automatic query steering framework for interactive data exploration”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD ’14)*. 2014, pp. 517–528 (cited on p. 12).

[45] W. Duivesteijn, A. J. Feelders, A. Knobbe. “Exceptional model mining”. In: *Data Mining and Knowledge Discovery* 30.1 (2016), pp. 47–98 (cited on p. 13).

[46] V. Dzyuba, M. van Leeuwen. “Interactive discovery of interesting subgroup sets”. In: *Proceedings of the 12th International Symposium on Intelligent Data Analysis (IDA ’13)*. 2013, pp. 150–161 (cited on pp. 8, 24, 25, 115, 135).

[47] V. Dzyuba, M. van Leeuwen. “Learning what matters – Sampling interesting patterns”. In: *Proceedings of the 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD ’17)*. 2017, pp. 534–546. arXiv: 1702.01975 (cited on pp. 9, 95, 135).

[48] V. Dzyuba, M. van Leeuwen, L. De Raedt. “Flexible constrained sampling with guarantees for pattern mining”. In: *Data Mining and Knowledge Discovery* (in press). arXiv: 1610.09263 (cited on pp. 9, 59, 135).

[49] V. Dzyuba, M. van Leeuwen, S. Nijssen, L. De Raedt. “Active preference learning for ranking patterns”. In: *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI ’13)*. 2013, pp. 532–539 (cited on pp. 8, 27, 135).

[50] V. Dzyuba, M. van Leeuwen, S. Nijssen, L. De Raedt. “Interactive learning of pattern rankings”. In: *International Journal on Artificial Intelligence Tools* 23.06 (2014) (cited on pp. 8, 27, 135).

[51] S. Ermon, C. P. Gomes, A. Sabharwal, B. Selman. “Embed and project: Discrete sampling with universal hashing”. In: *Advances in Neural Information Processing Systems 26*. 2013, pp. 2085–2093 (cited on p. 60).

[52] S. Ermon, C. P. Gomes, A. Sabharwal, B. Selman. “Taming the curse of dimensionality: Discrete integration by hashing and optimization”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML ’13)*. 2013, pp. 334–342 (cited on p. 92).

[53] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth. “From data mining to knowledge discovery in databases”. In: *AI magazine* 17.3 (1996), p. 37 (cited on p. 1).

[54] S. Filippi, O. Cappé, A. Garivier, C. Szepesvári. “Parametric bandits: The generalized linear case”. In: *Advances in Neural Information Processing Systems 23*. 2010, pp. 586–594 (cited on pp. 109, 115).

[55] Y. Freund, R. Iyer, R. E. Schapire, Y. Singer. “An efficient boosting algorithm for combining preferences”. In: *Journal of Machine Learning Research* 4 (2003), pp. 933–969 (cited on p. 37).

[56] E. Galbrun, P. Miettinen. “SIREN: An interactive tool for mining and visualizing geospatial redescriptions”. In: *Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’12)*. 2012, pp. 1544–1547 (cited on p. 20).

[57] F. Geerts, B. Goethals, T. Mielikäinen. “Tiling databases”. In: *Proceedings of the 7th International Conference on Discovery Science (DS ’04)*. 2004, pp. 278–289 (cited on p. 79).

[58] A. Gelman. “Exploratory data analysis for complex models”. In: *Journal of Computational and Graphical Statistics* 13.4 (2004), pp. 755–779 (cited on p. 117).

[59] L. Geng, H. Hamilton. “Interestingness measures for data mining: A survey”. In: *ACM Computing Surveys* 38.3 (2006) (cited on p. 16).

[60] A. Giacometti, D. H. Li, P. Marcel, A. Soulet. “20 years of pattern mining: A bibliometric survey”. In: *ACM SIGKDD Explorations Newsletter* 15.1 (2013), pp. 41–50 (cited on pp. 2, 14, 19).

[61] A. Giacometti, A. Soulet. “Anytime algorithm for frequent pattern outlier detection”. In: *International Journal of Data Science and Analytics* 2.3 (2016), pp. 119–130 (cited on p. 91).

[62] G. Gigerenzer, R. Selten. *Bounded rationality: the adaptive toolbox*. MIT Press, 2002 (cited on p. 116).

[63] A. Gionis, H. Mannila, T. Mielikäinen, P. Tsaparas. “Assessing data mining results via swap randomization”. In: *ACM Transactions on Knowledge Discovery from Data* 1.3 (2007), p. 14 (cited on p. 117).

[64] B. Goethals, S. Moens, J. Vreeken. “MIME: A framework for interactive visual pattern mining”. In: *Proceedings of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’11)*. 2011, pp. 757–760 (cited on p. 20).

[65] C. P. Gomes, W.-j. van Hoeve, A. Sabharwal, B. Selman. “Counting CSP solutions using generalized XOR constraints”. In: *Proceedings of the 21st AAAI Conference on Artificial Intelligence (AAAI ’07)*. 2007, pp. 204–209 (cited on pp. 71, 92).

[66] C. P. Gomes, A. Sabharwal, B. Selman. “Near-uniform sampling of combinatorial spaces using XOR constraints”. In: *Advances in Neural Information Processing Systems 19*. 2007, pp. 481–488 (cited on pp. 60, 67).

[67] T. Guns, S. Nijssen, L. De Raedt. “Evaluating pattern set mining strategies in a constraint programming framework”. In: *Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD ’11)*. 2011, pp. 382–394 (cited on p. 17).

[68] T. Guns, S. Nijssen, L. De Raedt. “Itemset mining: A constraint programming perspective”. In: *Artificial Intelligence* 175.12-13 (2011), pp. 1951–1983 (cited on pp. 16, 61, 62, 64, 65, 71).

[69] T. Guns, S. Nijssen, L. De Raedt. “k-pattern set mining under constraints”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.2 (2013), pp. 402–418 (cited on pp. 17, 64, 78, 92).

[70] T. Guns, S. Nijssen, A. Zimmermann, L. De Raedt. “Declarative heuristic search for pattern set mining”. In: *Proceedings of the 11th IEEE International Conference on Data Mining Workshops (ICDMW’ 11)*. 2011, pp. 1104–1111 (cited on p. 17).

[71] D. J. Hand, R. J. Bolton. “Pattern discovery and detection: A unified statistical methodology”. In: *Journal of Applied Statistics* 31.8 (2004), pp. 885–924 (cited on p. 115).

[72] M. A. Hasan, M. J. Zaki. “Output space sampling for graph patterns”. In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 730–741 (cited on pp. 18, 60, 61, 64, 98).

[73] E. Hüllermeier, J. Fürnkranz, eds. *Preference learning*. Springer, 2011, p. 454 (cited on pp. 4, 31).

[74] A. Jameson, B. Berendt, S. Gabrielli, C. Gena, F. Cena, F. Vernero, K. Reinecke. “Choice architecture for human-computer interaction”. In: *Foundations and Trends in Human-Computer Interaction* 7.1–2 (2014), pp. 1–235 (cited on p. 117).

[75] S. Jaroszewicz, T. Scheffer, D. Simovici. “Scalable pattern mining with bayesian networks as background knowledge”. In: *Data Mining and Knowledge Discovery* 18.1 (2008), pp. 56–100 (cited on pp. 19, 29).

[76] T. Joachims. “Optimizing search engines using clickthrough data”. In: *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’02)*. 2002, pp. 133–142 (cited on pp. 37, 99, 115).

[77] D. Kahneman, A. Tversky. “Subjective probability: A judgment of representativeness”. In: *Cognitive Psychology* 3.3 (1972), pp. 430–454 (cited on p. 116).

[78] T. Kamishima, H. Kazawa, S. Akaho. “A survey and empirical comparison of object ranking methods”. In: *Preference Learning*. Ed. by J. Fürnkranz, E. Hüllermeier. Springer, 2011. Chap. III, pp. 181–202 (cited on p. 31).

[79] A. Kemmar, W. Ugarte, S. Loudni, T. Charnois, Y. Lebbah, P. Boizumault, B. Crémilleux. “Mining relevant sequence patterns with cp-based framework”. In: *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence (ICTAI ’14)*. 2014, pp. 552–559 (cited on pp. 64, 92).

[80] M. Khiari, P. Boizumault, B. Crémilleux. “Constraint programming for mining n-ary patterns”. In: *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming (CP ’10)*. 2010, pp. 552–567 (cited on pp. 17, 64).

[81] J. Kleinberg, C. Papadimitriou, P. Raghavan. “A microeconomic view of data mining”. In: *Data mining and knowledge discovery* 2.4 (1998), pp. 311–324 (cited on p. 19).

[82] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, A. I. Verkamo. “Finding interesting rules from large sets of discovered association rules”. In: *Proceedings of the 3rd ACM International Conference on Information and Knowledge Management (CIKM ’94)*. 1994, pp. 401–407 (cited on p. 19).

[83] W. Klösgen. “Explora: A multipattern and multistrategy discovery assistant”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy. AAAI Press, 1996. Chap. 10, pp. 249–271 (cited on p. 13).

[84] B. P. Knijnenburg, N. J. Reijmer, M. C. Willemsen. “Each to his own: How different users call for different interaction methods in recommender systems”. In: *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys ’11)*. ACM. 2011, pp. 141–148 (cited on p. 117).

[85] A. Knobbe, B. Crémilleux, J. Fürnkranz, M. Scholz. “From local patterns to global models: The LeGo approach to data mining”. In: *Proceedings of the LeGo ECML/PKDD Workshop*. 2008 (cited on p. 17).

[86] A. Knobbe, E. Ho. “Pattern teams”. In: *Proceedings of the 10th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD ’06)*. 2006, pp. 577–584 (cited on pp. 17, 64).

[87] K.-N. Kontonasios, T. De Bie. “Formalizing complex prior information to quantify subjective interestingness”. In: *Proceedings of the 11th International Symposium on Intelligent Data Analysis (IDA ’12)*. 2012, pp. 161–171 (cited on p. 19).

[88] K.-N. Kontonasios, E. Spyropoulou, T. De Bie. “Knowledge discovery interestingness measures based on unexpectedness”. In: *WIREs: Data Mining and Knowledge Discovery* 2.5 (2012), pp. 386–399 (cited on p. 19).

[89] P. Kralj Novak, N. Lavrač, G. Webb. “Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining”. In: *Journal of Machine Learning Research* 10 (2009), pp. 377–403 (cited on p. 13).

[90] N. Lavrač, B. Kavšek, P. Flach, L. Todorovski. “Subgroup discovery with cn2-sd”. In: *Journal of Machine Learning Research* 5.Feb (2004), pp. 153–188 (cited on p. 17).

[91] Y. LeCun, Y. Bengio, G. Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444 (cited on p. 12).

[92] P. E. Lehner, L. Adelman, B. A. Cheikes, M. J. Brown. “Confirmation bias in complex analyses”. In: *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans* 38.3 (2008), pp. 584–592 (cited on p. 117).

[93] F. Lemmerich, M. Becker, F. Puppe. “Difference-based estimates for generalization-aware subgroup discovery”. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML/PKDD ’13)*. 2013, pp. 288–303 (cited on p. 92).

[94] H. Li. *Learning to rank for information retrieval and natural language processing*. 2nd. Morgan & Claypool, 2014, p. 121 (cited on p. 4).

[95] G. Liu, H. Zhang, M. Feng, L. Wong, S.-k. Ng. “Supporting exploratory hypothesis testing and analysis”. In: *ACM Transactions on Knowledge Discovery from Data* 9.4 (2015), pp. 1–24 (cited on p. 12).

[96] M. Albrecht, G. Bard. *The M4RI Library*. The M4RI Team. 2012 (cited on pp. 80, 92).

[97] M. Mampaey, J. Vreeken, N. Tatti. “Summarizing data succinctly with the most informative itemsets”. In: *ACM Transactions on Knowledge Discovery from Data* 6.4 (2012) (cited on p. 17).

[98] H. Mannila, H. Toivonen. “Multiple uses of frequent sets and condensed representations”. In: *Proceedings of the 2nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’96)*. 1996, pp. 189–194 (cited on p. 12).

[99] D. Martens, J. Vanthienen, W. Verbeke, B. Baesens. “Performance of classification models from a user perspective”. In: *Decision Support Systems* 51.4 (2011), pp. 782–793 (cited on p. 19).

[100] K. McGarry. “A survey of interestingness measures for knowledge discovery”. In: *The knowledge engineering review* 20.01 (2005), pp. 39–61 (cited on p. 16).

[101] K. Meel, M. Vardi, S. Chakraborty, D. Fremont, S. Seshia, D. Fried, A. Ivrii, S. Malik. “Constrained sampling and counting: Universal hashing meets SAT solving”. In: *Proceedings of the Beyond NP AAAI Workshop*. 2016 (cited on p. 60).

[102] P. Miettinen. “Interactive data mining considered harmful (if done wrong)”. In: *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics (IDEA ’14)*. 2014, pp. 85–87 (cited on p. 117).

[103] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, H. Mannila. “The discrete basis problem”. In: *IEEE Transactions on Knowledge and Data Engineering* 20.10 (2008), pp. 1348–1362 (cited on p. 17).

[104] B. Negrevergne, A. Dries, T. Guns, S. Nijssen. “Dominance programming for itemset mining”. In: *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM’ 13)*. 2013, pp. 557–566 (cited on p. 31).

[105] R. S. Nickerson. “Confirmation bias: A ubiquitous phenomenon in many guises”. In: *Review of general psychology* 2.2 (1998), pp. 175–220 (cited on p. 117).

[106] S. Nijssen, T. Guns, L. De Raedt. “Correlated itemset mining in ROC space: A constraint programming approach”. In: *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’09)*. 2009, pp. 647–655 (cited on p. 92).

[107] S. Nijssen, A. Zimmermann. “Constraint-based pattern mining”. In: *Frequent Pattern Mining*. Ed. by C. C. Aggarwal, J. Han. Springer, 2014. Chap. 7, pp. 147–163 (cited on pp. 16, 59).

[108] B. Padmanabhan, A. Tuzhilin. "A belief-driven method for discovering unexpected patterns". In: *Proceedings of the 4th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '98)*. 1998, pp. 332–336 (cited on p. 19).

[109] S. Paramonov, M. van Leeuwen, M. Denecker, L. De Raedt. "An exercise in declarative modeling for relational query mining". In: *Proceedings of the 25th International Conference on Inductive Logic Programming (ILP '15)*. 2015, pp. 166–182 (cited on p. 64).

[110] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal. "Discovering frequent closed itemsets for association rules". In: *Proceedings of the 7th International Conference on Database Theory (ICDT '99)*. 1999, pp. 398–416 (cited on p. 16).

[111] R. Pearson. *Exploring data in engineering, the sciences, and medicine*. Oxford University Press, 2011, p. 792 (cited on p. 11).

[112] J. Pei, J. Han. "Can we push more constraints into frequent pattern mining?" In: *Proceedings of the 6th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '00)*. 2000, pp. 350–354 (cited on pp. 16, 63).

[113] C. Powers. *Gartner: Companies will rely on citizen data scientists*. `https://www.asug.com/news/gartner-companies-will-rely-on-citizen-data-scientists`. (Online; accessed May 11, 2017). 2015 (cited on p. 5).

[114] F. Provost, T. Fawcett. "Data science and its relationship to big data and data-driven decision making". In: *Big Data* 1.1 (2013), pp. 51–59 (cited on p. 1).

[115] B. Qian, X. Wang, J. Wang, H. Li, N. Cao, W. Zhi, I. Davidson. "Fast pairwise query selection for large-scale active learning to rank". In: *Proceedings of the 13th IEEE International Conference on Data Mining (ICDM '13)*. 2013, pp. 607–616 (cited on p. 40).

[116] F. Radlinski, T. Joachims. "Active exploration for learning rankings from clickthrough data". In: *Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '07)*. 2007, pp. 570–579 (cited on p. 31).

[117] F. Radlinski, R. Kleinberg, T. Joachims. "Learning diverse rankings with multi-armed bandits". In: *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. 2008, pp. 784–791 (cited on p. 115).

[118] N. Ramakrishnan, D. Kumar, B. Mishra, M. Potts, R. Helm. "Turning CARTwheels: An alternating algorithm for mining redescriptions". In: *Proceedings of the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '04)*. 2004, pp. 266–275 (cited on p. 92).

[119] S. Rizzi. "What-if analysis". In: *Encyclopedia of Database Systems*. Springer, 2009, pp. 3525–3529 (cited on p. 115).

[120] F. Rossi, K. B. Venable, T. Walsh. *A short introduction to preferences: Between artificial intelligence and social choice*. Morgan & Claypool, 2011, p. 102 (cited on p. 117).

[121] S. Rueping. "Ranking interesting subgroups". In: *Proceedings of the 26th International Conference on Machine Learning (ICML '09)*. 2009, pp. 913–920 (cited on pp. 3, 6, 20, 23, 30).

[122] S. Sahar. "Interestingness via what is not interesting". In: *Proceedings of the 5th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '99)*. 1999, pp. 332–336 (cited on pp. 6, 20, 23).

[123] T. Sellam, M. Kersten. "Meet Charles, big data query advisor". In: *Proceedings of the 6th Biennial Conference on Innovative Data Systems Research (CIDR '13)*. 2013 (cited on p. 12).

[124] S. Shalev-Shwartz, A. Tewari. "Stochastic methods for ℓ_1-regularized loss minimization". In: *Journal of Machine Learning Research* 12 (2011), pp. 1865–1892 (cited on pp. 37, 100).

[125] X. Shen, C. Zhai. "Active feedback in ad hoc information retrieval". In: *Proceedings of the 28th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '05)*. 2005, pp. 59–66 (cited on pp. 31, 39).

[126] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, K. M. Borgwardt. "Efficient graphlet kernels for large graph comparison." In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS '09)*. 2009, pp. 488–495 (cited on p. 64).

[127] P. K. Shivaswamy, T. Joachims. "Coactive learning". In: *Journal of Artificial Intelligence Research* 53.1 (2015), pp. 1–40 (cited on pp. 31, 115).

[128] A. Silberschatz, A. Tuzhilin. "On subjective measures of interestingness in knowledge discovery". In: *Proceedings of the 1st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '95)*. 1995, pp. 275–281 (cited on pp. 2, 19).

[129] P. Slovic. "The construction of preference." In: *American Psychologist* 50.5 (1995), pp. 364–371 (cited on p. 116).

[130] M. Soos. “Enhanced gaussian elimination in DPLL-based SAT solvers”. In: *Proceedings of the Pragmatics of SAT Workshop (POS ’10)*. 2010, pp. 2–14 (cited on p. 74).

[131] N. Tatti, M. Mampaey. “Using background knowledge to rank itemsets”. In: *Data Mining and Knowledge Discovery* 21.2 (2010), pp. 293–309 (cited on p. 19).

[132] S. Teso, P. Dragone, A. Passerini. “Coactive critiquing: Elicitation of preferences and features”. In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI ’17)*. 2017, pp. 2639–2645 (cited on p. 31).

[133] S. Teso, A. Passerini, P. Viappiani. “Constructive preference elicitation by setwise max-margin learning”. In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI ’16)*. 2016, pp. 2067–2073 (cited on p. 31).

[134] J. W. Tukey. *Exploratory data analysis*. Pearson, 1977, p. 688 (cited on p. 11).

[135] A. Tversky, D. Kahneman. “The framing of decisions and the psychology of choice”. In: *Science* 211.4481 (1981), pp. 453–458 (cited on p. 116).

[136] T. Uno, M. Kiyomi, H. Arimura. “LCM ver. 3: Collaboration of array, bitmap and prefix tree for frequent itemset mining”. In: *Proceedings of the 1st International Workshop on Open Source Data Mining: Frequent Pattern Mining Implementations (OSDM ’05)*. 2005, pp. 77–86 (cited on p. 86).

[138] M. van Leeuwen. “Interactive data exploration using pattern mining”. In: *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*. Ed. by A. Holzinger, I. Jurisica. Vol. 8401. Springer, 2014, pp. 169–182 (cited on pp. 3, 19).

[139] M. van Leeuwen, L. Cardinaels. “VIPER–visual pattern explorer”. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML/PKDD ’15)*. 2015, pp. 333–336 (cited on p. 20).

[140] M. van Leeuwen, A. Knobbe. “Diverse subgroup set discovery”. In: *Data Mining and Knowledge Discovery* 25.2 (2012), pp. 208–242 (cited on pp. 13, 17, 24, 25, 43, 45).

[141] M. van Leeuwen, A. Ukkonen. “Discovering skylines of subgroup sets”. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML/PKDD ’13)*. 2013, pp. 273–287 (cited on pp. 17, 101).

[142] J. Vanschoren, H. Blockeel, B. Pfahringer, G. Holmes. "Experiment databases". In: *Machine Learning* 87.2 (2012), pp. 127–158 (cited on p. 116).

[143] J. Vreeken, M. van Leeuwen, A. Siebes. "Krimp: Mining itemsets that compress". In: *Data Mining and Knowledge Discovery* 23.1 (2010), pp. 169–214 (cited on pp. 17, 115).

[144] J. Vreeken, N. Tatti. "Interesting patterns". In: *Frequent Pattern Mining*. Ed. by C. Aggarwal, J. Han. Springer, 2014, pp. 65–81 (cited on p. 19).

[145] C. Wallace. *Statistical and inductive inference by minimum message length*. Springer, 2005, p. 429 (cited on p. 115).

[146] G. Webb, J. Vreeken. "Efficient discovery of the most interesting associations". In: *ACM Transactions on Knowledge Discovery from Data* 8.3 (2014), pp. 15–1 (cited on p. 16).

[147] H. Wickham. *How is data science different to mainstream statistics?* `http://blogs.lse.ac.uk/impactofsocialsciences/2014/09/23/data-science-statistics-communication/`. (Online; accessed March 21, 2017). 2014 (cited on p. 12).

[148] S. Wrobel. "An algorithm for multi-relational discovery of subgroups". In: *Proceedings of the 1st European Conference on Principles of Data Mining and Knowledge Discovery (PKDD '97)*. 1997, pp. 78–87 (cited on p. 13).

[149] X. Wu, V. Kumar, R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, D. Steinberg. "Top 10 algorithms in data mining". In: *Knowledge and information systems* 14.1 (2008), pp. 1–37 (cited on p. 1).

[150] D. Xin, X. Shen, Q. Mei, J. Han. "Discovering interesting patterns through user's interactive feedback". In: *Proceedings of the 12th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '06)*. 2006, pp. 773–778 (cited on pp. 3, 6, 20, 23, 30).

[151] Z. Xu, K. Kersting, T. Joachims. "Fast active exploration for link-based preference learning using gaussian processes". In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML/PKDD '10)*. 2010, pp. 499–514 (cited on p. 31).

[152] Z. Xu, R. Akella, Y. Zhang. "Incorporating diversity and density in active learning for relevance feedback". In: *Proceedings of the 29th European Conference on Information Retrieval (ECIR '07)*. 2007, pp. 246–257 (cited on pp. 31, 39).

[153] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li. “New algorithms for fast discovery of association rules”. In: *Proceedings of the 3rd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD ’97)*. 1997, pp. 283–296 (cited on pp. 7, 62, 69).

[154] A. Zimmermann, S. Nijssen. “Supervised pattern mining and applications to classification”. In: *Frequent Pattern Mining*. Ed. by C. C. Aggarwal, J. Han. Springer, 2014. Chap. 17, pp. 425–442 (cited on pp. 16, 59).

List of publications

Journal articles

V. Dzyuba, M. van Leeuwen, L. De Raedt. "Flexible constrained sampling with guarantees for pattern mining". In: *Data Mining and Knowledge Discovery* (in press). arXiv: 1610.09263

V. Dzyuba, M. van Leeuwen, S. Nijssen, L. De Raedt. "Interactive learning of pattern rankings". In: *International Journal on Artificial Intelligence Tools* 23.06 (2014)

Peer-reviewed conference papers

V. Dzyuba, M. van Leeuwen. "Learning what matters – Sampling interesting patterns". In: *Proceedings of the 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD '17)*. 2017, pp. 534–546. arXiv: 1702.01975

V. Dzyuba, M. van Leeuwen, S. Nijssen, L. De Raedt. "Active preference learning for ranking patterns". In: *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence (ICTAI '13)*. 2013, pp. 532–539 **Best Paper Award** (out of 299 submissions).

V. Dzyuba, M. van Leeuwen. "Interactive discovery of interesting subgroup sets". In: *Proceedings of the 12th International Symposium on Intelligent Data Analysis (IDA '13)*. 2013, pp. 150–161

Conference papers not covered in this dissertation

T. Decroos, V. Dzyuba, J. Van Haaren, J. Davis. "Predicting soccer highlights from spatio-temporal match event streams". In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI '17)*. 2017, pp. 1302–1308

J. Van Haaren, V. Dzyuba, S. Hannosset, J. Davis. "Automatically discovering offensive patterns in soccer match data". In: *Proceedings of the 14th International Symposium on Intelligent Data Analysis (IDA '15)*. 2015, pp. 286–297

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DECLARATIVE LANGUAGES AND ARTIFICIAL INTELLIGENCE
Celestijnenlaan 200A box 2402
B-3001 Leuven
vladimir.dzyuba@kuleuven.be
https://dtai.cs.kuleuven.be/